inubit Suite 6

inubit Workbench/Process Engine

Systemkonnektor-Guide

Copyright

inubit AG

Schöneberger Ufer 89-91
10785 Berlin
Deutschland

Phone: +49.30.72 61 12-0
Fax: +49.30.72 61 12-100
E-Mail: contact@inubit.com
URL: www.inubit.com

Inhaltsverzeichnis

Zielgruppe

Der Systemkonnektor-Guide enthält detaillierte Informationen für Systemadministratoren, Systemintegratoren und Entwickler, welche die inubit Suite 6 (oder Teile davon) verwalten und konfigurieren.

Umfang der Dokumentation

Die inubit Suite 6 bietet eine umfassende Dokumentation, die Ihnen als gedrucktes Handbuch, als PDF-Datei und als Onlinehilfe in inubit Workbench zur Verfügung steht.

Die Dokumentation besteht aus folgenden Teilen:

- **inubit Suite 6**
  - **Quick Start**
    Beschreibt die Hard- und Softwarevoraussetzungen, die Installation und die ersten Schritte.
  - **Migrationsanleitung**
  - **Tutorials**
    Für den Ein- und Durchstieg. Die Tutorials erläutern die Verwendung der wichtigsten Komponenten der inubit Suite 6 anhand von fachlichen Szenarien.
- **inubit Workbench, inubit Process Engine** und **inubit Enterprise Portal**:
  - **Benutzer-Guide**
    Beschreibt das Arbeiten mit inubit Workbench, das Erstellen der verschiedenen Diagrammtypen und Module, das Arbeiten mit Metadaten und Workflow Variablen, Simulationen, Tests, das fachlich orientierte Monitoring und Reporting.
  - **Administrator- und Entwickler-Guide**
    Enthält administrative Themen wie Konfiguration der inubit Process Engine, Backup und Restore, Benutzerverwaltung, Security-Aspekte, Monitoring und Clustering, Entwicklung von Plug-ins und Thin Clients.
  - **Modul-Guide**
    Data Converter, Format Adapter, Utilities, Workflow und Web Service Controls verwenden und konfigurieren.
  - **Systemkonnektor-Guide**
    Alles über Einsatz und Konfiguration von Systemkonnektoren.
- **inubit Solution Center 3.0**
  - **Benutzer-Guide**
    Informationen, wie Sie Fachmodelle anlegen, mit Business Solutions arbeiten, Ansichten erstellen, Prozesse und Standardmodelle einbinden sowie eine Dokumentation der REST-Schnittstelle.

- **Administrator-Guide**
  Erläutert, wie Sie Daten sichern und wiederherstellen, das inubit Solution Center Server als Dienst installieren, Ports anpassen, das Portal, die Datenbank und HTTPS konfigurieren, Benutzer verwalten und Diagramme importieren.

- **inubit WebModeler 2.3**
  - **Administrator- und Benutzer-Guide**
    Alles über das Anlegen und Bearbeiten von Modellen, Anpassen der Ports, Konfigurieren der Datenbank und von HTTPS.

Die aktuelle Dokumentation steht im inubit User-Portal im Register „Software“ unter der Adresse https://www.inubit-user.com zum Download bereit.

Weitere Informationen

Die folgenden Informationen liegen als Booklet der DVD bei bzw. sind als Dateien im Installationspaket enthalten:

- **readme.txt**
  Hinweise zur Installation und Migration der inubit Suite 6.

Lesen Sie diese Datei grundsätzlich vor der Installation oder Aktualisierung der inubit Suite 6!

- **Quick Start**
  Systemvoraussetzungen und Installationsanleitung als Booklet der inubit Suite 6-DVD.
- **API-Dokumentation des Plug-in Software Development Kits**
  Im Verzeichnis <iS-installdir>/documentation/apidoc/index.html.
- **JavaScript-Framework**
  Im Verzeichnis `<iS-installdir>/documentation/jsdoc/index.html`.

Tipps, Hinweise und Links in der Dokumentation

Tipps bieten nützliche Informationen für das Arbeiten mit der inubit Suite 6.

Hinweise sollten Sie unbedingt lesen und beachten. Das Nichtbeachten kann den Verlust von Daten oder schwerwiegende Systemprobleme verursachen.

→ Verweise auf eine andere Textstelle in der Dokumentation der inubit Suite 6 sind mit einem Pfeil gekennzeichnet.

Links auf Webseiten erkennen Sie an dem nebenstehenden Symbol.

Weitere Informationen und Support

Pressemitteilungen und Whitepapers stehen auf unserer Website www.inubit.com für Sie zum Download bereit.

Für weitere Informationen über die inubit Suite 6 oder bei Fragen zum Einsatz der inubit Suite 6 nutzen Sie folgende Kontaktmöglichkeiten:

- **inubit-Support**: www.inubit.com/support
- **E-Mail**: support@inubit.com
- **Telefon**: +49.30.72 61 12-112

Viel Erfolg beim Arbeiten mit der inubit Suite 6 wünscht Ihnen das inubit Team!

1 Systemkonnektoren

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Systemkonnektoren verbinden Quell- bzw. Zielapplikationen mit der inubit Process Engine und leiten Nachrichten ohne Transformation durch.

In einem Systemkonnektor werden die spezifischen Protokoll- und Authentifizierungsdaten für den Aufbau der Verbindung mit den Quell- bzw. Zielapplikationen gespeichert.

→ Siehe auch

- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*
- *Systemkonnektoren aktivieren (Workbench: Benutzer-Guide, Kap. 3.9, S. 125)*
- *Systeme integrieren und Prozesse automatisieren (Tutorials, Kap. 6, S. 107)*

1.1 Dialogbeschreibungen

Dieser Abschnitt erläutert die folgenden Themen:

1.1.1 Dialog „Allgemeine Moduleigenschaften“

→ Siehe *Dialog „Allgemeine Moduleigenschaften“ (Workbench: Benutzer-Guide, Kap. 3.15, S. 130).*

1.1.2 Dialog „System Connector Eigenschaften“

Dieser Dialog bietet folgende Optionen:

Grundkonfiguration

- **Connector Typ**:
  Für Systemkonnektoren gibt es folgende Konfigurationstypen. Es gibt Systemkonnektoren, bei denen nicht alle Typen konfigurierbar sind. In diesen Fällen sind die Optionen deaktiviert.
  - **Input Connector** (Workflow-Anfang)
    Als Input Connector holt ein Systemkonnektor Daten von einer Quellapplikation und übergibt diese an das Nachfolgemodul. Nach der Workflowausführung sendet der Input Connector das Ergebnis wieder an die Quellapplikation zurück.
  - **Medium Connector** (Workflow-Mitte)
    Ein Medium Connector setzt in der Regel einen synchronen Aufruf an eine Applikation ab, nimmt die Antwort der Applikation entgegen und leitet diese das Nachfolge-Modul im Workflow weiter.
  - **Output Connector** (Workflow-Ende)
    Ein Output System Connector sendet Daten an eine Zielapplikation.

  → Der Konnektortyp wird am Modul durch ein Symbol signalisiert. Eine Übersicht der Symbole finden Sie im Abschnitt *Symbole in Technical Workflow und im Verzeichnisbaum (Workbench: Benutzer-Guide, Kap. 12, S. 332)*.
- **Auf Datenempfang warten (Listener Connector)**
  Es sind nicht alle Systemkonnektoren als Listener konfigurierbar. In diesen Fällen ist die Option deaktiviert.
  Ein Systemkonnektor kann als Listener betrieben werden:
  - Input Connector Listener warten auf Aufrufe oder Daten aus einer Quellapplikation.
  - Output Connector Listener stellen Daten für ein Zielsystem zum Abholen bereit.

Connector Status

- **Aktiv/Inaktiv**
  Damit Systemkonnektoren in Technical Workflows oder BPEL-Diagrammen zeitgesteuert und automatisiert ausgeführt werden, müssen diese aktiv sein. Inaktive Systemkonnektoren können nur manuell gestartet werden, z. B. zum Testen.

  → Siehe *Systemkonnektoren aktivieren (Workbench: Benutzer-Guide, Kap. 3.9, S. 125).*

Remote Connector

Wenn markiert, dann verbindet sich der Systemkonnektor zu einem Remote Connector auf einem entfernten Rechner.

Es kann nicht jeder Systemkonnektortyp als Remote Connector genutzt werden. Diese Option ist nur aktiv, wenn der Systemkonnektor als Remote Connector genutzt werden kann.

Systemkonnektoren, die als Listener konfiguriert sind, können grundsätzlich nicht als Remote Connector betrieben werden.

Der Button „Konfiguration“ öffnet den *Dialog „Remote Connector Konfiguration“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1.1.5, S. 25)*. In diesem Dialog konfigurieren Sie die URL des Remote Connectors und Authentifizierungsdaten und können die Verbindung testen.

→ Für mehr Informationen über die Installation und Konfiguration eines Remote Connectors siehe *Remote Connector (Process Engine: Administrator- und Entwickler-Guide, Kap. 13, S. 143)*.

Behandlung von Verbindungsfehlern

- **Anzahl der Wiederholversuche**
  Gibt an, wie oft der Systemkonnektor versuchen soll, eine Verbindung zu dem anzusprechenden System aufzubauen.
  Wenn die Verbindung nicht wiederhergestellt werden kann, wird ein Fehler geworfen.
- **Zeitabstand der Wiederholversuche**
  Zeitintervall zwischen den einzelnen Versuchen, eine Verbindung mit dem System herzustellen.
- **Timeout (in Sek.)/Standard-Timeout**
  Zeit, die das Modul auf die Herstellung der Verbindung wartet. Wenn nach Ablauf dieser Zeit keine Verbindung hergestellt werden kann, dann wird die Modulausführung abgebrochen und ein Fehler geworfen.
  Wenn Sie den Wert auf Null setzen, ist die Wartezeit unendlich. Wenn Sie die Option „Standard-Timeout“ markieren, wird der Timeout auf 600 Sekunden gesetzt.

Input/Output Message Logging

Wenn diese Option aktiv ist, werden die Eingabe- und Ausgangsnachrichten des Systemkonnektors gespeichert.

Diese Funktion ist nützlich bei der Fehlersuche und beim Testen, wenn z. B Nachrichten nicht bei einem Geschäftspartner angekommen sind und erneut übermittelt werden müssen, oder wenn ein Workflow fehlerhaft ausgeführt wurde und mit denselben Eingangsnachrichten erneut gestartet werden soll. Workflows starten Sie im Queue Manager neu. Dort können Sie auch die Eingabe- und Ausgangsnachrichten anzeigen lassen.

→ Siehe

- *Message Log anzeigen (Process Engine: Administrator- und Entwickler-Guide, Kap. 7.6.4, S. 105)*
- *Prozess erneut starten (Process Engine: Administrator- und Entwickler-Guide, Kap. 7.6.5, S. 105)*

Aktivieren Sie das Input/Output Message Logging nur, wenn Sie es benötigen. Sonst kann bei wenig Plattenplatz das Dateisystem mit Dateien aus dem Input/Output Message Logging voll laufen.

Wenn das Input/Output Logging bei mehreren Systemkonnektoren und/oder im Technical Workflow selbst aktiviert ist, dann wird die kleinste Angabe verwendet!

Sobald das Input/Output Message Logging aktiviert ist, können Sie festlegen, wie viele Nachrichten aufbewahrt werden sollen:

- **Anzahl Messages**
  Zum Festlegen der Anzahl der Dateien an, die aufbewahrt werden soll. Falls diese Anzahl im Verlauf der Workflow-Verarbeitung überschritten wird, dann werden die ältesten Nachrichten gelöscht, bis die vorgegebene Anzahl wieder erreicht ist.
- **Synchronisiere mit System Log**
  Wenn diese Option markiert ist, dann werden die Nachrichten so behandelt, wie bei der Option „System Logging“ im Diagramm angegeben.
  → Siehe *System-Logging (Workbench: Benutzer-Guide, Kap. , S. 109)*..
  Wenn die Option „System Logging“ nicht markiert ist, dann werden die Standard-Einstellungen verwendet, die in der Datei `<iS-installdir>\server\ibis_root\conf\logsDBConfig.xml` im Element `dataEntriesLimit` festgelegt sind.

1.1.3 Dialog „Zeitgesteuerte Verarbeitung“

Systemkonnektoren können zeitgesteuert ausgeführt werden. In diesem Dialog legen Sie Startzeiten oder Startintervalle fest. Systemkonnektoren starten nur zeitgesteuert, wenn der dazu gehörige Technical Workflow auf eine inubit Process Engine publiziert und aktiviert wurde. Im lokalen Modus und im Test-Modus werden zeitgesteuerte Systemkonnektoren nicht gestartet.

Zeitplan

- **Scheduler-Aktivierung**

Wenn die Option markiert ist, dann verhalten sich Systemkonnektoren folgendermaßen:

- **Input Connector**

  Startet zum angegebenen Zeitpunkt bzw. im angegebenen Intervall und stößt damit die Verarbeitung des Technical Workflows an.
- **Output Connector**

  Sendet Nachrichten zum definierten Zeitpunkt bzw. Intervall an die Zielapplikation. Bis dahin werden die Nachrichten zwischengespeichert.

Wenn ein Systemkonnektor in einem Intervall gestartet wird, das kürzer ist als die Ausführung des Workflows (z. B. Intervall=10 s, Ausführung=45 s), dann verhält sich der Workflow folgendermaßen:

- Wenn der Workflow parallel geschaltet ist, dann wird er im angegebenen Intervall gestartet. Falls die maximale Anzahl möglicher, paralleler Ausführungen, erreicht ist, dann wird die weitere Bearbeitung in die Queue gestellt.
- Wenn der Workflow nicht parallelisiert ist und zum Zeitpunkt seines nächsten Aufrufs noch läuft, dann wird der Workflow nicht erneut gestartet.

Tage

Ein Systemkonnektor kann folgendermaßen aktiviert werden:

- täglich,
- wöchentlich (Auswahl eines oder mehrerer Wochentage
- monatlich (Auswahl eines oder mehrere Tage eines Monats)

Uhrzeit

Startzeit, zu welcher der Prozess an dem angegeben Tag gestartet werden soll. Wenn ein Intervall aktiviert ist, dann wird der Vorgang in einem anzugebenden zeitlichen Abstand erlaubt (z. B. alle 5 Minuten).

Um das Wiederholen des Vorgangs zu beenden, muss ein Endzeitpunkt definiert werden.

Zeitraum

Zeitraum (Anfangs- und Enddatum), in dem der Scheduler aktiviert werden soll.

Leseverhalten von Input-Konnektoren

- Bei aktivierter Zeitsteuerung werden Workflows mit folgenden Input-Konnektoren so oft neu gestartet wie Daten vorhanden sind:
  - Mail Connector
  - inubit iS Connector
  - JMS Connector

- Bei aktivierter Zeitsteuerung kann für folgende Input-Konnektoren die maximale Anzahl von Ausführungen pro zeitgesteuertem Aufruf definiert werden:
  - File Connector
    → Siehe *Dialog „Zu lesende Datei(en)“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 10.2.1, S. 122).*
  - VFS Connector
    → Siehe *Dialog „Input Connector-Konfiguration“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 38.2.2, S. 392).*
  - FTP Connector
    → Siehe *Dialog „Input-Dateinamen definieren“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 11.2.3, S. 136).*

Leseverhalten von Medium-/ Output-Konnektoren

Konnektoren, die ein Vorgängermodul haben, werden bei jedem Workflow-Durchlauf genau einmal ausgeführt.

Behandlung von Fehlern

Bei jedem fehlerhaften Aufruf Log-Eintrag erstellen

Wenn aktiviert, dann wird für jeden Fehler, der beim Start des Systemkonnektors auftritt, ein Eintrag im Register „Monitoring > System Log“ erzeugt.

1.1.4 Dialog „SSL-Konfiguration“

Dieser Dialog bietet folgende Optionen zum Festlegen der SSL-Einstellungen beim Verbindungsaufbau des Systemkonnektors:

- **Server-Authentifizierung**

  Wenn markiert, dann muss sich der Server, zu dem die Verbindung aufgebaut wird, gegenüber dem Systemkonnektor identifizieren.

  Wenn nicht markiert, dann wird jeder SSL-Server als vertrauenswürdig betrachtet.

  Der Button „Truststore laden“ öffnet einen Dateiexplorer zum Laden des öffentlichen Schlüssels bzw. Truststores des Servers.

  Als Truststore wird das Java Keystore-, PEM- oder DER-Format akzeptiert.
- **Client-Authentifizierung**

  Wenn markiert, dann muss sich der Systemkonnektor gegenüber dem Server identifizieren.

Der Button „Keystore laden“ öffnet einen Dateiexplorer zum Laden des privaten Schlüssels bzw. Keystores des Servers.

Der Keystore kann im Java Keystore- oder PEM-Format vorliegen.

Um den Keystore laden zu können, müssen Sie das Passwort für den privaten Schlüssel bzw. den Keystore eingeben!

→ Die Server- und Client-Authentifizierung wird im Anmeldedialog der inubit Workbench und bei folgenden Systemkonnektoren angeboten:

- *inubit IS Connector (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 14, S. 153)*
- *FTP Connector (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 11, S. 131)*
- *HTTP Connector (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 13, S. 145)*
- *AS2 Connector (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 3, S. 31)*
- *RosettaNet HTTPS Connector (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 30, S. 307)*

1.1.5 Dialog „Remote Connector Konfiguration“

→ Für mehr Informationen über Installation und Konfiguration eines Remote Connectors siehe *Remote Connector (Process Engine: Administrator- und Entwickler-Guide, Kap. 13, S. 143)*.

- **URL**

  URL des Remote Connectors.

  Ersetzen Sie `localhost` durch den Namen des Rechners, auf dem der Remote Connector installiert ist, oder durch dessen IP-Adresse.

- **Server-Authentifizierung**

  Wenn markiert, dann muss sich der Remote Connector auf dem entfernten Rechner gegenüber dem Systemkonnektor, den Sie gerade konfigurieren, mit einem Zertifikat identifizieren.

  Wenn nicht markiert, dann wird jeder SSL-Server als vertrauenswürdig betrachtet.

  Der Button „Datei auswählen“ öffnet einen Dateiexplorer zum Laden des öffentlichen Schlüssels bzw. Truststores des Remote Connectors.

  Nach den Laden wird das Datum angezeigt, bis zu dem der Schlüssel gültig ist.

- **Client-Authentifizierung**

Wenn markiert, dann muss sich der Systemkonnektor, den Sie gerade konfigurieren, gegenüber dem entfernten Remote Connector mit einem privaten Schlüssel bzw. Keystore identifizieren.

Der Button „Datei auswählen“ öffnet einen Dateiexplorer zum Laden des privaten Schlüssels bzw. Truststores.

Nach den Laden wird das Datum angezeigt, bis zu dem der Schlüssel gültig ist.

Verbindungstest

- **Verbindung testen**

  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

Dieser Abschnitt erläutert die folgenden Themen:

■ *Beispiel für eine XPCML-Eingangsnachricht, S. 27*
■ *Dialog „AS/400 Connector Eigenschaften“, S. 28*

Verwendung

Mit dem AS/400 Connector können Sie Anwendungen auf einem AS/400-System aus einem Technical Workflow heraus starten.

Zum Aufrufen der Anwendung und Übergeben evtl. vorhandener Parameter benötigen Sie eine XPCML-basierte Eingangsnachricht. Diese Eingangsnachricht erstellen Sie mit Hilfe eines XSLT Converters, der vor den AS/400 Connector geschaltet wird.

→ Siehe auch
- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

2.1 Beispiel für eine XPCML-Eingangsnachricht

Eine XPCML-basierte Eingangsnachricht enthält Eingangs-, Durchgangs- und Ausgangsparameter inkl. Datentypen und Werten.

Der AS/400 Connector erwartet folgende Eingaben:

■ Typ, Wert und Länge aller erforderlichen Parameter
■ Vollständiger Pfad zu der AS/400 Anwendung

Beispiel

Zur Überprüfung der Vertragsnummern werden folgende Parameter übergeben:

■ Parameter 1 = Mandant (char, 1-stellig, Ausprägung 1=AnbieterX,2=Anbieter2)
■ Parameter 2 = Vertragsnummer (char, 17-stellig)
■ Parameter 3 = return code (char, 1-stellig, Ausprägung)
■ Auszuführendes Programm „vertragsNrAbfrage“

Anfrage in XPCML

```
<?xml version="1.0" encoding="UTF-8"?>
<xpcml
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
  xsi:noNamespaceSchemaLocation="xpcml.xsd"
  version="4.0">
  <program name="vertragsNrAbfrage"
    path="/XYZ.lib/myprog.lib
      /vertragsNrAbfrage.pgm">
    <parameterList>
      <stringParm name="mandant"
        passDirection="inout" length="1">
          2
      </stringParm>
      <stringParm name="vertragsNr"
        passDirection="inout" length="17">
          110
      </stringParm>
      <stringParm name="vertragsNrBekannt"
        passDirection="inout" length="1">
          0
      </stringParm>
    </parameterList>
  </program>
</xpcml>
```

2.2 Dialog „AS/400 Connector Eigenschaften“

Dieser Dialog bietet folgende Optionen:

Einstellungen

- **Adresse**
  Servername oder IP-Adresse des AS/400 Systems.
- **Benutzer**
  Benutzername für den Zugang des AS/400 Systems.
- **Passwort**
  Passwort für das AS/400 System.
- **Programmname**
  Name des Programms (ohne Erweiterung), das gestartet werden soll.
- **Erweiterter Debugmode**

Wenn markiert, dann wird die Debug-Ausgabe des AS/400 Systems zu den Debug-Ausgaben der inubit Process Engine hinzugefügt.

Verbindungstest

- **Verbindung testen**

  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

3 AS2 Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Der AS2 Connector versendet und empfängt Geschäftsnachrichten und Empfangsbestätigungen gemäß dem Applicability Statement 2 (AS2).

AS2 beschreibt, wie Geschäftsnachrichten sicher über das HTTP-Protokoll in einem Peer-to-Peer-Netzwerk ausgetauscht werden. AS2 definiert, wie die Verbindung hergestellt, Geschäftsnachrichten validiert, versendet und durch Empfangsbestätigungen bestätigt werden.

Die Urheberschaft einer Geschäftsnachricht wird durch digitale Signaturen und die Datensicherheit durch Verschlüsselung gewährleistet. Der Absender erhält aus dem AS2-Übertragungsstandard eine digitale Empfangsquittung (Message Disposition Notification, MDN), mit welcher der Absender die fristgerechte Zustellung beweisen kann.

Vor dem Versenden wird für jede Geschäftsnachricht ein Umschlag in Form einer Standard-MIME-Struktur erzeugt.

Die Geschäftsnachrichten können in einem XML-Format, EDI-Format (z. B. ANSI X12 oder UN/EDIFACT) oder in jedem anderen strukturierten Format vorliegen.

Siehe IETF-AS2 Spezifikation unter http://www.ietf.org/rfc/rfc4130.txt

Der AS2 Connector wurde erfolgreich mit folgenden Produkten getestet:

- Compinia ComAS2
- OpenAS2
- IP*Works AS2 Connector

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

3.1 Modulvariablen des AS2 Connectors

Variablen des AS2 Connectors

Die Tabelle enthält alle Variablen, welche ein AS2 Connector bei seiner Ausführung setzt, unabhängig davon, ob es sich um einen Input oder Output Connector handelt. Die Variablen stehen im weiteren Ablauf des Workflows zur Verfügung:

Name	Wert
`AS2Subject`	Wert des HTTP-Headers „Subject“ der Eingangsnachricht.
`ASMessageMDNOptions`	Wert des HTTP-Headers für „Disposition-Notification-Options" der Eingangsnachricht.
`ASMessageMIC`	Wert der berechneten Prüfsumme der Eingangsnachricht.
`ASMessageMICALG`	Name des Algorithmus, mit dem die Prüfsumme berechnet wurde.

Variablen des AS2 Input Connectors

Die folgende Tabelle enthält die Variablen, die ein AS2 Input Connector bei seiner Ausführung setzt:

Name	Wert
`ASMessageID`	ID der eingehenden Nachricht
`ASMessageTo`	AS2-ID des Empfängers
`ASMessageFrom`	AS2-ID des Absenders
`ASMessageError`	Fehlermeldung, Fehler können z.B. sein: ■ authentication-failed: Signaturprüfung der eingehenden Nachricht ist fehlgeschlagen ■ decryption-failed: Entschlüsselung ist fehlgeschlagen ■ sender-equals-receiver. Empfänger ist mit dem Absender identisch Für weitere Fehlermeldungen siehe die AS2 Spezifikation.
`ASMessageDate`	Zeitstempel des Empfangs im Format „EEE, d MMM yyyy HH:mm:ss Z“, dabei bedeutet: E = Werktag (Montag bis Sonntag) d = Tag des Monats (1 bis 31) y = Jahr (0 bis 9) H = Stunde des Tages (0 bis 23) m = Minute (0 bis 59) s = Sekunde (0 bis 59) Z = Zeitzone (Zeitzonenangabe gemäß RFC 822)
`ASMessageFailure`	Fehlermeldung. Siehe AS2 Spezifikation
`ASMessageWarning`	Warnung. Siehe AS2 Spezifikation

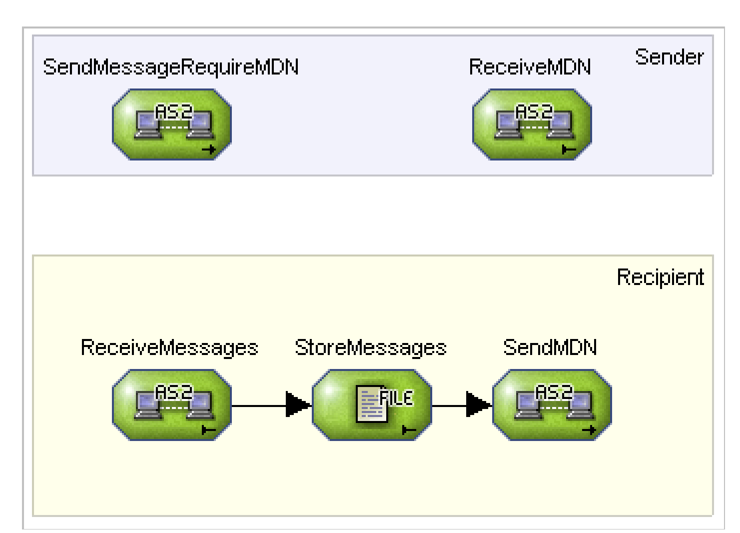
→ Für Infos über Variablen und deren Verwendung siehe *Workflow-Variablen und Mappings (Workbench: Benutzer-Guide, Kap. 14, S. 365)*.

3.2 Empfangen, Senden und Prüfen von AS2-Nachrichten

AS2 bietet einen gesicherten Datentransport über die so genannte sichere Übertragungsschleife (secure transmission loop) mit asymmetrischer Verschlüsselung. Dabei werden die Nachrichten verschlüsselt und signiert. Zusätzlich können Empfangsbestätigungen angefordert werden.

Zum Verschlüsseln und Signieren besitzen Absender und Empfänger einer Nachricht jeweils ein Schlüsselpaar bestehend aus einem öffentlichen und einem privaten Schlüssel und tauschen ihre öffentlichen Schlüssel miteinander aus. Beide besitzen also den öffentlichen Schlüssel des anderen und jeweils einen eigenen privaten Schlüssel, der nie herausgegeben wird.

Die folgenden Workflows illustrieren das Versenden und Empfangen einer AS2-Nachricht und ihrer Empfangsbestätigung (Message Disposition Notification, MDN):



Die Module sind folgendermaßen konfiguriert:

1. **`SendMessagesRequireMDN`**
   Der Output Connector verschlüsselt die zu übertragenden Daten mit dem öffentlichen Schlüssel des Empfängers, sendet diese ab und fordert eine asynchrone MDN als Bestätigung, dass die Nachricht angekommen ist. Die Nachricht enthält u. a.

- eine ID, die in einem Verzeichnis gespeichert wird, weil sie noch zur Identifikation der MDN benötigt wird,
- die Adresse, an welche die MDN gesendet werden soll,
- eine Prüfsumme über den Nachrichten-Inhalt.

2. **ReceiveMessages**

   Der Input Listener Connector wartet auf den Eingang der Nachricht. Sobald diese eintrifft, entschlüsselt er diese mit seinem privaten Schlüssel und startet den Workflow.

3. **StoreMessages**

   Der File Connector speichert die Nachricht beim Empfänger

4. **SendMDN**

   Der Output Connector erstellt automatisch die MDN, wenn eine angefordert wurde.

   Die MDN enthält die ID der Nachricht, eine Prüfsumme über den Nachrichteninhalt und eine Signatur.

   Die Signatur wurde mit dem privaten Schlüssel des Nachrichten-Empfängers erstellt und garantiert, dass die Bestätigung wirklich vom Nachrichten-Empfänger kommt. ID, Adresse und Prüfsumme werden über Variablen automatisch vom Input zum Output Connector übertragen.

   Der Output Connector schickt die MDN an den Absender zurück. Die MDN wird automatisch entsprechend den Anforderungen des Absenders (hier: Output Connector `SendMessagesRequireMDN`) synchron oder asynchron zurück gesendet.

5. **ReceiveMDN**

   Der Input Listener wartet auf die MDN. Sobald diese eintrifft, wird die darin enthaltene ID in demselben Verzeichnis gespeichert, in dem die ID bereits beim Versenden der Nachricht gespeichert wurde. Die ID wird genutzt, um die Daten der dazugehörigen, im Dateisystem abgespeicherten AS2-Nachricht zu finden.

   Die Prüfsummen werden miteinander verglichen, um sicherzustellen, dass die Nachricht vollständig und korrekt beim Empfänger angekommen ist.

   Zusätzlich kann der Absender mit dem öffentlichen Schlüssel des Empfängers die Korrektheit der Signatur überprüfen.

   Der Input Listener erzeugt eine XML-Ausgangsnachricht mit folgender Struktur:

```
<?xml version="1.0" encoding="UTF-8"?>
<MDNReport>
 <MDNSignature>true</MDNSignature>
 <MIC>true</MIC>
 <Error>decryption-failed</Error>
</MDNReport>
```

   Dabei haben die Elemente folgende Bedeutung:

   - `<MDNSignature>`:

Signatur der Empfangsbestätigung

- true, wenn die Signatur korrekt ist
- false, wenn die Empfangsbestätigung nicht signiert wurde oder wenn die Signaturprüfung einen Fehler ergeben hat

- `<MIC>`
  (Message Integrity Check) True oder false, abhängig davon, ob die Prüfsummen über den Inhalt übereinstimmen.
- `<Error>`
  Platzhalter für folgende Fehler-Elemente:
  - Failure, Error oder Warning: Enthalten eine kurze Fehlermeldung, die in der AS2-Spezifikation beschrieben ist.
  - FailureMessage, ErrorMessage oder WarningMessage: Enthalten ausführliche Fehlermeldungen (selten unterstützt).

Siehe IETF-AS2 Spezifikation unter http://www.ietf.org/rfc/rfc4130.txt

3.3 Dialogbeschreibungen

Dieser Abschnitt erläutert die folgenden Themen:

3.3.1 Dialog „AS2 Konfiguration“

(Input Listener Connector)

In diesem Dialog legen Sie fest, ob der Input Listener Connector Nachrichten oder Empfangsbestätigungen empfangen soll.

3.3.2 Dialog „AS S/MIME Konfiguration“

(Nachrichtenempfang mit Input Listener)

Dieser Dialog bietet folgende Optionen:

Signatur überprüfen

Wenn die Option markiert ist, dann werden die Signaturen aller eingehenden Nachrichten überprüft. Zum Prüfen wird der öffentliche Schlüssel des Geschäftspartners benötigt, von dem Sie die Nachrichten empfangen.

- **Truststore oder Zertifikat hinzufügen** (Button)
  Öffnet einen Dateiexplorer zum Laden des öffentlichen Schlüssels.
  Nach dem Laden wird über dem Button die Gültigkeit des Schlüssels angezeigt.
- **Eingehende Nachricht muss signiert sein**
  Wenn markiert, dann erzeugt das Eintreffen einer unsignierten Nachricht einen Fehler.

Entschlüsselung

Wenn die Option markiert ist, dann werden alle eingehenden Nachrichten entschlüsselt. Dazu benötigen Sie Ihren privaten Schlüssel.

- **Keystore hinzufügen** (Button)
  Öffnet einen Dateiexplorer zum Laden des Schlüssels. Nach dem Laden wird über dem Button die Gültigkeit des Schlüssels angezeigt.
- **Eingehende Nachricht muss verschlüsselt sein**
  Wenn markiert, dann erzeugt das Eintreffen einer unverschlüsselten Nachricht einen Fehler.

3.3.3 Dialog „AS2 Listener Konfiguration“

(Nachrichtenempfang mit Listener)

Dieser Dialog bietet folgende Optionen:

Konfiguration des Servers

- **URL**
  URL des Servers, auf dem die inubit Process Engine mit dem AS2 Input Connector läuft. Die URL ist standardmäßig vorbelegt mit einer URL nach folgendem Muster:
  `http://<localhost:8000>/ibis/servlet/IBISHTTPUploadServlet/<Module-Name>`
  Passen Sie ggf. die URL an.
- **Authentifizierung**
  Wenn für Ihren Server eine Authentifizierung notwendig ist, geben Sie hier die entsprechenden Daten an.

3.3.4 Dialog „Message Archiv Konfiguration“

(MDN-Empfang mit Input Listener)

→ Siehe *Modulvariablen des AS2 Connectors (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 3.1, S. 32).*

3.3.5 Dialog „AS MDN Validator“

(Output Connector/MDN-Empfang mit Input Listener)

→ Siehe *Modulvariablen des AS2 Connectors (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 3.1, S. 32).*

3.3.6 Dialog „AS2 Listener Konfiguration“

(MDN-Empfang mit Listener/Nachrichten senden mit Output Connector)

In diesem Dialog geben Sie den Server an, an den die eingehenden AS2 Nachrichten geschickt werden sollen.

Konfiguration des Servers

- **URL**
  Gibt die Adresse des Servers an, z. B http://www.example.com
- **Authentifizierung**

Wenn für den Zugang zum Server eine Authentifizierung notwendig ist, markieren Sie die Checkbox und geben Login und Passwort an, das Sie von Ihrem Geschäftspartner für diesen Server erhalten haben.

3.3.7 Dialog „AS2 Konfiguration“

(Output Connector)

Im Dialog zur AS2 Output Konfiguration legen Sie fest, ob der Output Connector Nachrichten oder Empfangsbestätigungen senden soll.

3.3.8 Dialog „AS Nachrichten Konfiguration“

(Nachrichtenversand mit Output Connector)

Dieser Dialog bietet folgende Optionen:

- **Absender/Empfänger (AS ID)**
  Geben Sie die Absender und Empfänger ID ein (Zeichenketten mit max. 128 Zeichen). Diese ID müssen Sie vor dem ersten Nachrichten-Austausch mit Ihrem Geschäftspartner vereinbaren.
- **Betreff**
  Aussagekräftige Angabe zu der Nachricht.
- **Kodierung**
  Wählen Sie eine Kodierung aus, die der Kodierung Ihrer AS2 Nachrichten entspricht.

Um binäre Daten (z. B. Multimedia-Daten) zu übertragen, wählen Sie den leeren Eintrag.

- **MIME Typ**
  Wählen Sie einen MIME-Typ aus, welcher dem Typ Ihrer AS2-Nachrichten entspricht. Wenn der von Ihnen benötigte MIME-Typ nicht vorhanden ist, geben Sie diesen ein.

Siehe http://www.iana.org/assignments/media-types/ für eine Liste aller MIME Media Typen.

- **Dateiname**
  Geben Sie optional einen Namen für den Nachrichteninhalt der AS2-Nachricht an. Dieser wird als `filename`-Parameter in den HTTP-Header `Content-Disposition` übernommen.

3.3.9 Dialog „AS/MIME Konfiguration“

(Nachrichtenversand mit Output Connector)

In diesem Dialog legen Sie fest, ob Nachrichten signiert, verschlüsselt und/oder komprimiert werden.

Signieren

Wenn markiert, dann werden alle ausgehenden Nachrichten mit Ihrem privaten Schlüssel signiert.

- **Gültig bis**:
  Zeigt nach dem Laden des Schlüssels dessen Gültigkeitsdatum an.
- Button **Keystore hinzufügen**
  Öffnet einen Dateiexplorer zum Laden des Schlüssels.

Verschlüsseln

Wenn markiert, dann werden alle ausgehenden Nachrichten mit dem öffentlichen Schlüssel Ihres Geschäftspartners verschlüsselt.

Wenn Ihr Geschäftspartner zum Empfang von AS2-Nachrichten nicht den AS2 Connector der inubit Suite 6 verwendet, dann müssen Sie vor dem Verschlüsseln den Verschlüsselungsalgorithmus seiner Software ermitteln, weil nicht alle Produkte alle Algorithmen unterstützen.

- **Verschlüsselungsalgorithmus**
  Zur Auswahl des Verschlüsselungsalgorithmus. Standard ist „DES_EDE3_CBC“, weil dieses Verfahren das gängigste ist.
  Die Empfängerseite erkennt das Verschlüsselungsverfahren der Nachricht automatisch.
- **Gültig bis**:
  Zeigt nach dem Laden des Schlüssels das Gültigkeitsdatum an.
- Button **Truststore oder Zertifikat hinzufügen**
  Öffnet einen Dateiexplorer zum Laden des öffentlichen Schlüssels.

Komprimieren

Wenn markiert, dann werden die Nachrichten entsprechend der S/MIME-Spezifikation komprimiert.

3.3.10 Dialog „AS MDN Konfiguration“

(Nachrichtenversand mit Output Connector)

In diesem Dialog legen Sie fest, ob Sie Empfangsbestätigungen (MDN) vom Empfänger erwarten. Standardmäßig werden synchrone Empfangsbestätigungen verlangt.

Synchrone Empfangsbestätigungen sendet der Empfänger sofort nach Erhalt der Nachrichten über HTTP an die Adresse, die in der Nachricht als Absenderadresse angegeben ist. Der Output Connector wartet solange, bis die Empfangsbestätigung eingetroffen ist. Um die Empfangsbestätigung zu speichern, schalten Sie z. B einen File Connector hinter den AS2 Output Connector.

Alternativ können Sie explizit angeben, dass Sie asynchrone und/oder signierte Empfangsbestätigungen erwarten:

- **Signierte Empfangsbestätigung erforderlich**
  Falls diese Option aktiviert ist, muss der Empfänger die zurückgesendete Empfangsbestätigung signieren und der MDN-Listener prüft die Signatur.
- **Asynchrone MDN**
  Asynchrone Empfangsbestätigungen werden zeitlich unabhängig von der Nachricht versendet. Zum Empfangen benötigen Sie einen Workflow mit einem AS2 Input Listener.
  Geben Sie die URL an, an welche die Empfangsbestätigung geschickt werden soll. Die URL wird zusammen mit der Nachricht unverschlüsselt über HTTP versendet.

3.3.11 Dialog „HTTP(S) Server Konfiguration“

(Nachrichtenversand mit Output Connector)

In diesem Dialog geben Sie die Einstellungen des Servers an, an den die AS2 Nachrichten geschickt werden sollen.

Konfiguration des Servers

- **URL/SSL**
  - Adresse des Servers, z. B http://www.example.com
  - Zum Konfigurieren der Server- bzw. Client-Authentifizierung. Öffnet den *Dialog „SSL-Konfiguration“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1.1.4, S. 24)*.
- **Authentifizierung**

Wenn für den Zugang zum Server eine Authentifizierung notwendig ist, markieren Sie die Checkbox und geben Login und Passwort an, das Sie von Ihrem Geschäftspartner für diesen Server erhalten haben.

→ Siehe *SSL-Verbindung und Server-Authentifizierung konfigurieren (Process Engine: Administrator- und Entwickler-Guide, Kap. 5.1, S. 55).*

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

3.3.12 Dialog „AS MDN Versand-Konfiguration“

(MDN-Versand mit Output Connector)

→ Siehe *Modulvariablen des AS2 Connectors (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 3.1, S. 32).*

Zum Laden des Keystores benötigen Sie dessen Alias und Passwort!

4 Backup Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Der Backup Connector sichert den aktuellen Systemzustand oder ausgewählte Benutzer und Benutzergruppen und deren Daten.

Der Backup Connector kann zeitgesteuert gestartet werden, so dass die Daten in regelmäßigen Zeitabständen automatisch gesichert werden können.

→ Siehe auch

- *Datensicherung -/Wiederherstellung (Process Engine: Administrator- und Entwickler-Guide, Kap. 10, S. 125)*
- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

4.1 Dialog „Backup Connector Eigenschaften“

Dieser Dialog bietet folgende Optionen:

Backup Dateiname

- **Name**
  Name der Sicherungsdatei.
  Die Sicherungsdatei wird in das Verzeichnis `<iS-install-dir>\server\ibis_root\ibis_data\backup` geschrieben.
- **In den Ausgabestrom schreiben**
  Wenn markiert, dann werden die gesicherten Daten zusätzlich als Ausgangsnachricht dem Workflow übergeben.

Backup Level

- **System**
  Sichert den aktuellen Systemzustand der inubit Process Engine. Folgende Verzeichnisse werden immer gesichert:
  - iS-Konfiguration (`<iS-installdir>/server/ibis_root/conf`, außer den EDI-Regeln im `edi`-Verzeichnis)

  - inubit Process Engine Log-Verzeichnis `<iS-install-dir>\server\ibis_root\log`
  - Die Benutzergruppe admin (umfasst alle Benutzer)
- **Benutzer/Benutzergruppen**
  - Um ausgewählte Benutzer bzw. Benutzergruppen zu sichern. Es werden alle Workflows, Module, Benutzer sowie Untergruppen und (optional) deren Repository- und Monitoring-Dateien gesichert.
  - Um alle Benutzergruppen zu sichern, markieren Sie die Gruppe „admin“.

    In diesem Fall wartet Backup Connector bis alle Workflows ihre aktuellen Aufgaben beendet haben.

Erweiterte Backup Optionen

- **iS Log Datenbank**
  - Systembackup:

    Die gesamte Logging-Datenbank wird gesichert, inkl. Status und Timeouts der Prozesse (`<iS-installdir>/server/ibis_root/log`).
  - Benutzer/Benutzergruppen-Backup:

    Die Daten der ausgewählten Benutzer/der ausgewählten Benutzergruppen werden gesichert.
- **Repository**
  - Systembackup:

    Es werden alle Daten unterhalb von `Global` gesichert.
  - Benutzer/Benutzergruppen-Backup:

    Die Daten der ausgewählten Benutzer/der ausgewählten Benutzergruppen werden gesichert.
- **Tasks & Waitings**

  Nur bei Systembackup:

  Sichert alle Tasks aller Benutzer aus der Task-Datenbank und dem Task-Verzeichnis `<iS-install-dir>\server\ibis_root\tasks` sowie die Nachrichten und Variablen aus wartenden Prozessen, die z. B. an Wait-Modulen und Multiplexern anliegen.

  → Für Informationen über das Wechseln der iS-Task-Datenbank siehe *iS-Task-Datenbank austauschen (Process Engine: Administrator- und Entwickler-Guide, Kap. 8.3, S. 116)*.
- **Portal**

  Sichert alle Organisationen und Communities des Portalservers, die im Bereich „Portal“ ausgewählt sind.

Portal

Es werden alle Organisationen und Communities angezeigt, die im konfigurierten Portalserver vorhanden sind. Bei einem Systembackup inkl. Portal werden alle markierten Organisationen und Communities gesichert.

5 Business Object Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Der Business Object Connector (BO Connector) ermöglicht es, XML-basierte Geschäftsobjekte, die in Ihren Technical Workflows verarbeitet werden, in einer relationalen Datenbank zu verwalten.

Der BO Connector übernimmt das Speichern und Laden der Geschäftsobjekte in der Datenbank transparent für den Entwickler und erzeugt die geeigneten SQL-Statements für die gewählte Datenbank. Damit erfordert ein Wechsel der Datenbank keine Änderungen am Technical Workflow mehr, es reicht, eine andere Datenbank im Modulassistenten zu konfigurieren.

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

5.1 Funktionsprinzip

Der BO Connector bildet die Strukturen der Geschäftsobjekte auf die relationalen Strukturen der Datenbank ab. Für dieses Mapping benötigt der BO Connector ein XML Schema, in dem die Strukturen und Beziehungen der XML-basierten Geschäftsobjekte beschrieben sind.

Das XML Schema kann entweder über das HTTP-Protokoll angesprochen oder im Repository gespeichert werden.

Beim ersten Hinzufügen eines Geschäftsobjekte werden Tabellen für das Geschäftsobjekte entsprechend dem XML-Schema in der Datenbank erzeugt.

Objekt- und Querymodus

Sie können den BO Connector in folgenden Modi nutzen:

- **Objektmodus**
  Die auszuführende Operation wird im BO Connector festgelegt. Das Geschäftsobjekte kann direkt als komplexes XML-Element an den BO Connector übergeben werden.
  → Siehe *Objektmodus verwenden (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 5.2, S. 48).*
- **Querymodus**
  Im Querymodus können Sie bei jeder Ausführung des Konnektors mehrere und unterschiedliche Typen von Operationen ausführen lassen.
  Die auszuführenden Operationen sind in der Eingangsnachricht des Konnektors definiert.
  → Siehe *Querymodus verwenden (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 5.3, S. 51).*

Objekt-Identifikatoren

Mit dem Objekt-Identifikator wird das gewünschte Geschäftsobjekt in der Datenbank adressiert.

5.2 Objektmodus verwenden

Überblick

Im Objektmodus wird immer die Operation ausgeführt, die im Modulassistenten des BO Connectors definiert ist.

→ Siehe Option *Operation (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 5, S. 56)*

5.2.1 Create/Update/Delete-Operationen

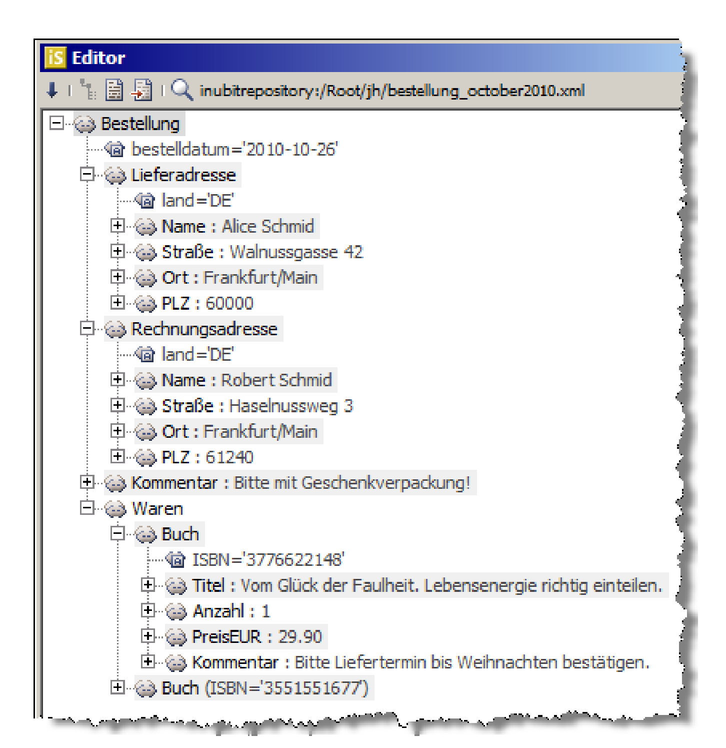
Bei Create-, Update- und Delete-Operationen erwartet der BO Connector ein XML-basiertes Geschäftsobjekt als Eingangsnachricht und gibt die Daten des Objekts und Namensraum-Angaben aus. Standardmäßig werden auch die Objekt-IDs ausgegeben, dieses Verhalten können Sie abschalten.

→ Siehe *Standard-ID ausgeben (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 5, S. 56)*

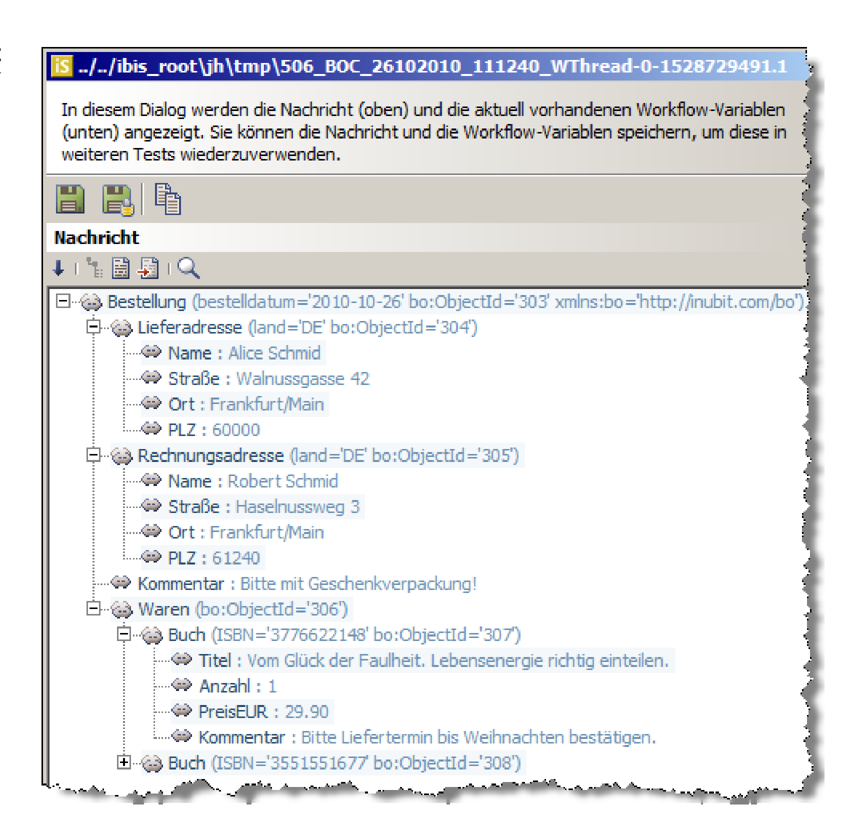
Voraussetzungen

Die Struktur des Geschäftsobjekts ist konform zu dem XML Schema, das im BO Connector definiert ist.

Beispiel für Eingangsnachricht



Beispiel für Ausgangsnachricht



5.2.2 Read-Operationen

Bei Read-Operationen erwartet der BO Connector als Eingangsnachricht das root-Element und die ID eines Geschäftsobjekts und gibt die Details des Geschäftsobjekts inkl. der Namensraum-Angaben aus.

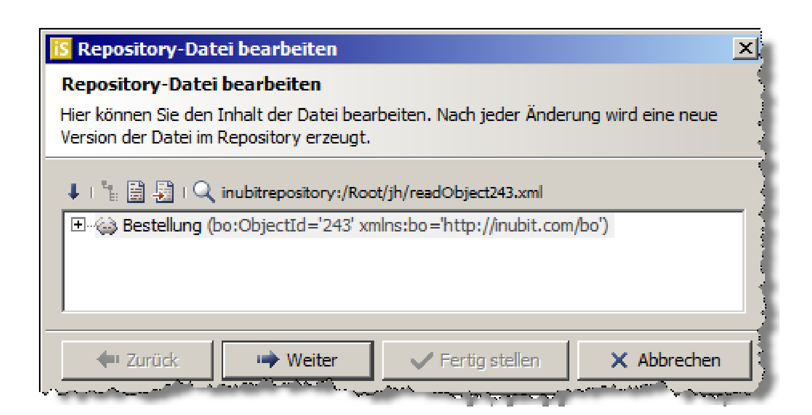
Standardmäßig werden auch die Objekt-IDs ausgegeben, dieses Verhalten können Sie abschalten.

→ Siehe *Standard-ID ausgeben (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 5, S. 56)*.

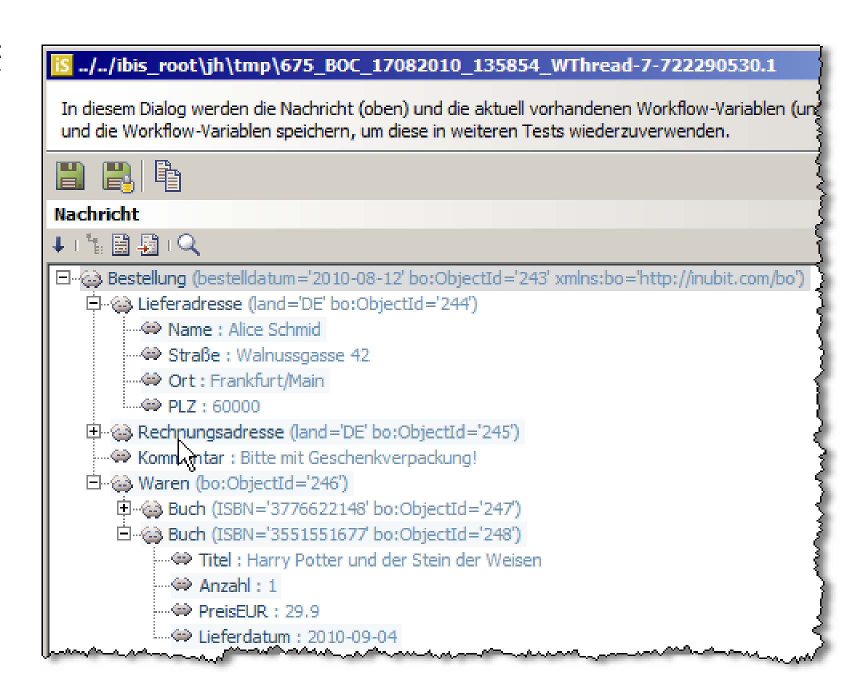
Voraussetzungen

Die Struktur des Geschäftsobjekts ist konform zu dem XML Schema sein, das im BO Connector definiert ist.

Beispiel für Eingangsnachricht



Beispiel für Ausgangsnachricht



5.3 Querymodus verwenden

Dieser Abschnitt erläutert die folgenden Themen:

- *Unterstützte Query-Attribute (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 5, S. 53)*

Im Querymodus werden die Operationen ausgeführt, die in der Eingangsnachricht des BO Connectors definiert sind.

Zum Erstellen der Eingangsnachricht benötigen Sie einen XSLT Converter: Der XSLT Converter erstellt die Eingangsnachricht auf Basis eines Mapping Templates und übergibt diese an den BO Connector.

Pfad des Mapping Templates

Das Mapping Template finden Sie im Repository unter „`Global > System > Mapping Templates > Business Object Connector > BO_Queries.xml`“.

Operationen definieren

In jeder Eingangsnachricht können mehrere Operationen definiert werden, auch Operationen mit unterschiedlichen Typen, wie z. B. `CREATE` und `DELETE`.

Die Operationen werden in den `type`-Attributen der `query`-Elemente definiert. Falls kein `type`-Attribut explizit angegeben ist, dann wird ein `READ` ausgeführt.

Beispiel für Eingangsnachricht

Mit der folgenden Eingangsnachricht werden erst alle Bestellungen aus der Datenbank gelesen und danach gelöscht.

```
<?xml version="1.0" encoding="UTF-8"?>
<queries>
  <!-- READ (default, if no type is defined) -->
  <query xmlns:bo="http://inubit.com/bo">
    /Bestellung
  </query>
  <!-- DELETE -->
  <query xmlns:bo="http://inubit.com/bo"
    type="delete">
    /Bestellung
  </query>
</queries>
```

Voraussetzungen

Sie haben eine XML-Beispielnachricht, die Ihr Geschäftsobjekt repräsentiert. Die Struktur des Geschäftsobjekts muss dem XML Schema entsprechen, das im Modulassistenten des BO Connectors angegeben ist.

So gehen Sie vor

1. Erstellen Sie einen XSLT Converter.
2. Laden Sie das Mapping Template in den Bereich „XML-Zieldatei“:
   a. Öffnen Sie das -Menü und wählen Sie „Öffnen > Repository…“. Der Repository Browser öffnet sich.

b. Navigieren Sie zu dem Mapping Template und laden Sie dieses.

   → Siehe *Pfad des Mapping Templates (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 5, S. 52)*

3. Ziehen Sie das `Queries`-Element unterhalb der gewünschten Operation nach oben in den Bereich „XSLT-Stylesheet“ auf das `template`-Element.
4. Um zusätzlich eine andere Operation durchführen zu lassen, fügen Sie ein query-Element mit dem gewünschten Attribut hinzu.
5. Laden Sie die XML-Beispielnachricht in den Bereich „XML-Quelldatei“.
6. Abhängig vom gewünschten Ergebnis ersetzen Sie die `MyObject`-Elemente des Templates durch die `root`-Elemente Ihrer Geschäftsobjekte oder geben einen XPath-Ausdruck an.

   Erläuterungen zu den Attributen finden Sie im Mapping Template.
7. Testen Sie die XSLT-Datei.

   → Siehe *XSLT-Stylesheets testen (Workbench/Process Engine: Modul-Guide, Kap. 5.2, S. 142).*
8. Wenn der Test erfolgreich war, können Sie den XSLT Converter publizieren.
9. Legen Sie einen BO Connector an.
10. Verbinden Sie den XSLT Converter mit Ihrem BO Connector.

5.3.1 Unterstützte Query-Attribute

Query-Attribute definieren Sie wie folgt: `<Attribut>="<Wert>"`, z. B. `type="create"`.

Die unterstützten Attribute für die einzelnen Operation entnehmen Sie dem Mapping Template im Abschnitt *Pfad des Mapping Templates (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 5, S. 52)*.

Attribut	Beschreibung	Mögliche Werte
type	Bestimmt die Operation der Query.	■ create: Objekt erstellen ■ read: Objekt lesen (Standard) ■ update: Objekt ändern ■ delete: Objekt löschen
queryId	Legt die Query-Id fest, welche am Result-set zur expliziten Prüfung ausgegeben wird.	Query-Id

Attribut	Beschreibung	Mögliche Werte
outputObjects	Verhindert die Ausgabe von Objekten im Resultset.	■ true: Objekte ausgeben (Standard) ■ false; Objekte nicht ausgeben
outputIds	Unterdrückt die Ausgabe der technischen Standard-Id "bo:ObjectId".	■ true: Objekt-Id ausgeben (Standard) ■ false: Objekt-Id nicht ausgeben, z. B. zur Web Service-Ansteuerung
commitCount	Legt das Commit-Intervall für Batch-Operationen fest.	■ 50: Commit nach 50 Objekten ■ 25: Commit nach 25 Objekten ■ 0: Commit am Ende der Modulausführung (Standard) ■ autoCommit: Commit nach jeder Operation
pageNum	Paging: Legt die zu lesende Seite in Bezug auf den im Attribut „PageSize“ angegebenen Wert fest.	■ 0: kein Paging (Standard) ■ positiver ganzzahliger Wert
PageSize	Legt die Anzahl der Einträge pro Seite beim Paging fest.	■ 0: kein Paging (Standard) ■ positiver ganzzahliger Wert
order	Legt die Sortierung innerhalb des Resultsets fest. Bei selektierten ComplexTypes wird nach bo:ObjectId, bei selektierten SimpleTypes nach dem SimpleType-Wert sortiert. Das Attribut „order“ kann auch mit einer expliziten Elementliste genutzt werden.	■ Asc: aufsteigend ■ Desc: absteigend ■ Attribut nicht gesetzt: Keine Sortierung
orderBy	Legt die Sortierung innerhalb des Resultsets mit expliziten Elementen des ComplexTypes fest.	■ Element ■ Sortierung Mehrere Sortierungen werden mit Kommas getrennt. Beispiel: orderBy="MyElement1 Asc, @bo:ObjectId Desc"
outputJSON	Alle Elemente innerhalb des Resultsets werden in JSON-Notation ausgegeben.	■ true: JSON-Notation, z. B. für AJAX-Responses ■ false; XML-Notation (Standard)
outputPureJSON	Die gesamte Ausgabe des BO Connectors erfolgt in JSON-Notation. **Hinweis**: Dieses Attribut muss am Root-Element angegeben werden!	■ true: JSON-Notation, z. B. für AJAX-Responses ■ false: XML-Notation (Standard)

<table>
<tr><th>Attribut</th><th>Beschreibung</th><th>Mögliche Werte</th></tr>
<tr><td>zipOutput</td><td>Das gesamte Resultset wird mittels GZIP komprimiert.
Hinweis: Dieses Attribut muss am Root-Element angegeben werden!</td><td>■ true: komprimiertes Resultset, z. B. zum Backup.
Zum Wiederherstellen des Backups kann der BO Connector das komprimierte Resultset einlesen.
■ false: unkomprimiertes Resultset (Standard)</td></tr>
</table>

5.4 Stammdaten aktualisieren und löschen

Stammdaten, wie Adressen oder Kundendaten, werden durch die Kardinalität `0..1` oder `1..1` definiert und gesondert behandelt. Wenn Sie bei einer Delete- oder Update-Operation für ein Geschäftsobjekt kein Stammdatum angeben, wird das Stammdatum nicht automatisch gelöscht oder aktualisiert, sondern nur die Referenz vom Geschäftsobjekt zum Stammdatum gelöscht oder aktualisiert.

5.5 Dialog „Business Object Connector Einstellungen"

Dieser Dialog bietet folgende Optionen:

Persistenzschicht-Definition

- **XML Schema**
  Das Schema wird benötigt, um Datenbanktabellen für die Geschäftsobjekte zu erstellen.
  Geben Sie die HTTP-URL ein, unter der das XML Schema zu erreichen ist oder klicken Sie auf [...], um ein XML Schema aus dem Repository zu laden.
- **Objekt-Identifikatoren**
  Die Spalte „Identifikator" dient zur Auswahl eines Elements im XML Schema, mit dem die Geschäftsobjekt eindeutig identifiziert werden können.

Objektmodus

- **Operation**
  Zur Auswahl der Operation, die der BO Connector im Objektmodus ausführen soll. Folgende Operationen sind verfügbar:
  - Create: Fügt ein Geschäftsobjekt hinzu.
  - Read: Liest ein Geschäftsobjekt.
  - Update: Ändert Details eines Geschäftsobjekt.
  - Delete: Löscht ein Geschäftsobjekt.

Im Querymodus wird die ausgewählte Operation ignoriert! Siehe auch *Objekt- und Querymodus (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 5, S. 48)*

- **Standard-ID ausgeben**
  Die Standard-ID benötigen Sie im Objektmodus, um auf ein bestimmtes Objekt zugreifen zu können (nur relevant im Objektmodus).
  Wenn die Option nicht markiert ist, werden keine IDs ausgegeben.

Datenbankverbindung

- **Voreinstellung**
  Wählen Sie den Typ der vorkonfigurierten Datenbank, um die Felder „Dialekt“, „Datenbank-URL“ und „Datenbanktreiber-Klasse“ sinnvoll zu belegen.
- **Dialekt**
  Wählen Sie den für Ihre Datenbankversion geeigneten Hibernate-Dialekt.
- **Datenbank-URL**
  Wird vorbelegt, wenn Sie eine vorkonfigurierte Datenbank gewählt haben.
  Geben Sie den Host, den Port und den Datenbanknamen an.
- **Benutzer/Passwort**
  Eingabe ist abhängig von Ihrer Datenbank.
- **Connection Pooling**
  Aktiviert das Connection Pooling.
  Wenn das Connection Pooling aktiviert ist, werden physische Verbindungen zu einer Datenbank wiederverwendet. Dies beschleunigt die Abarbeitung der Abfragen. Die Verbindungen werden nach ihrer Verwendung nicht abgebaut, sondern gespeichert und sind für nachfolgende Zugriffe verfügbar. So muss nicht bei jedem Zugriff auf die Datenbank erneut zeitintensiv eine Verbindung aufgebaut werden.
  Der Button „Einstellungen“ öffnet den *Dialog „Datenbank Connection Pooling“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 6.5.5, S. 87)* zur Konfiguration des Connection Poolings.

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

6 Database Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Der Database Connector verbindet sich über eine JDBC- oder ODBC-Schnittstelle mit einer Datenbank und bietet folgende Funktionen an:

- Metadatenabfragen
- Statische Datenbankabfragen via SQL
- Dynamische Datenbankabfragen via XML-Querys

In Stored Procedures können keine CLOBs übergeben werden! Der Database Connector unterstützt jedoch das Senden/Speichern von Daten als CLOB-Datentyp bei Oracle 9.* Datenbanken.

Konnektortypen

- **Input Connector**
  Sendete eine statische SQL-Query oder eine Metadatenabfrage an eine Datenbank und übergibt das Ergebnis an das nachfolgende Modul.
- **Medium/Output Connector**
  Sendet eine statische SQL-Query, eine Metadatenabfrage oder eine dynamische XML-Query an eine Datenbank, nimmt das Ergebnis der Datenbank entgegen und reicht es an das nächste Modul weiter.
  Dynamisch erzeugte XML-Querys sind z. B. sinnvoll, wenn die zu modifizierenden Daten erst zur Laufzeit bekannt werden.

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

6.1 Metadaten abfragen

Um Informationen über eine Datenbank zu erhalten, erstellen Sie eine Metadatenabfrage. Damit können Sie z. B. für jede Spalte aller Tabellen oder für jede Spalte einer ausgewählten Tabelle Infos über den Datentyp, die Spaltenbreite und zulässige Dezimalstellen erhalten.

So gehen Sie vor

1. Legen Sie einen Technical Workflow mit einem Database Connector an.
2. Aktivieren Sie beim Database Connector im *Dialog „Datenbankverbindungen und Query-Typ“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 6.5.1, S. 81)* im Bereich „Funktion“ die Option „Metadaten lesen“.
3. Klicken Sie auf „Weiter“. Der *Dialog „Metadaten“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 6.5.2, S. 84)* wird angezeigt.
4. Legen Sie fest, welche Metadaten zu welchen Tabellen ausgelesen werden sollen.

   Um Ihre Festlegungen zu prüfen, können Sie in dem Dialog eine Vorschau auf das Ausgabeformat der Metadaten anzeigen lassen.
5. Beenden Sie den Assistenten mit „Fertig stellen“.
6. Publizieren und aktivieren Sie den Technical Workflow.

6.2 Statische SQL-Query erstellen

Voraussetzungen

Sie benötigen die Metadaten Ihrer Datenbank, mindestens Tabellen- und Spaltennamen müssen bekannt sein.

→ Siehe *Metadaten abfragen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 6.1, S. 60).*

So gehen Sie vor

1. Legen Sie einen Technical Workflow mit einem Database Connector an.
2. Aktivieren Sie beim Database Connector im *Dialog „Datenbankverbindungen und Query-Typ“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 6.5.1, S. 81)* im Bereich „Funktion“ die Option „Statische Query ausführen“.

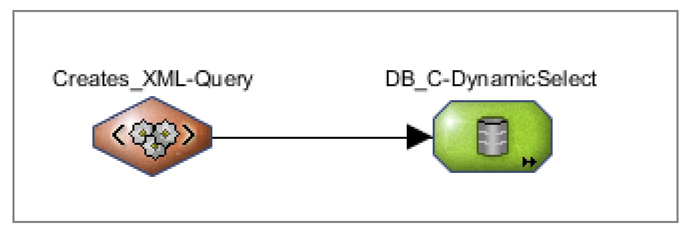
3. Klicken Sie auf „Weiter". Der *Dialog „Statische SQL-Query" (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 6.5.3, S. 85)* wird angezeigt.
4. Geben Sie die SQL-Query ein.
5. Klicken Sie auf „Weiter". Der *Dialog „Ergebnis der Query" (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 6.5.4, S. 85)* wird angezeigt.
6. Legen Sie fest, wie das Ergebnis der SQL-Query ausgegeben werden soll.
7. Beenden Sie den Assistenten mit „Fertig stellen".
8. Publizieren und aktivieren Sie den Technical Workflow.

Bei der Ausführung des Database Connectors wird die SQL Query an die Datenbank gesendet.

6.3 Dynamische XML-Query erstellen

Überblick

Um XML-Querys zur Ausführungszeit dynamisch zu erzeugen, benötigen Sie einen XSLT Converter, den Sie mit Ihrem Database Connector verbinden:



Der XSLT Converter konvertiert die XML-basierte Eingangsnachricht mit Hilfe eines XSLT-Stylesheets in eine XML-Query und übergibt die XML-Query zur Ausführung an den Database Connector.

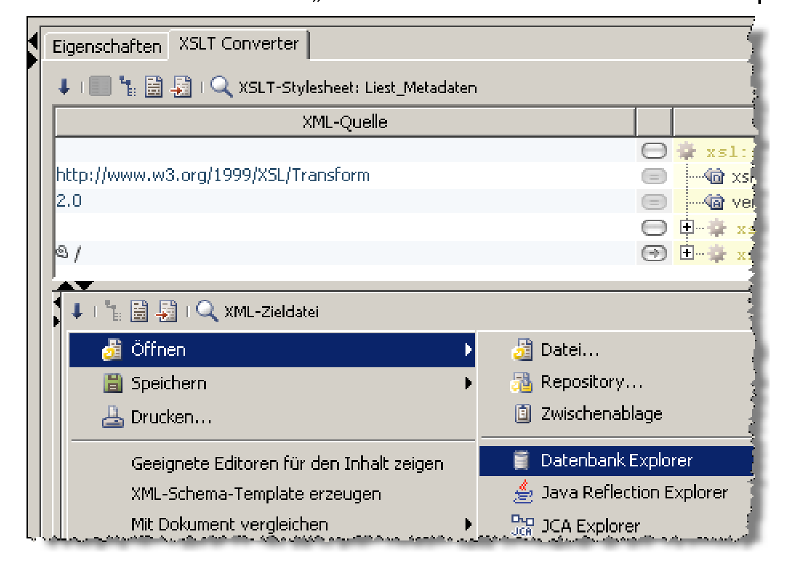
Voraussetzungen

Folgende Daten liegen vor:

- Metadaten Ihrer Datenbank, mindestens Tabellen- und Spaltennamen müssen bekannt sein.
  → Siehe *Metadaten abfragen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 6.1, S. 60).*
- Beispiel-XML-Eingangsnachricht.

So gehen Sie vor

1. Legen Sie einen Technical Workflow mit einem Database Medium oder Output Connector an.
2. Aktivieren Sie beim Database Connector im *Dialog „Datenbankverbindungen und Query-Typ“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 6.5.1, S. 81)* im Bereich „Funktion“ die Option „Eingangsnachricht als Query ausführen“.
3. Publizieren Sie den Database Connector.
4. Fügen Sie einen XSLT Converter in Ihren Technical Workflow ein.
5. Verbinden Sie die beiden Module.
6. Öffnen Sie den XSLT Converter zum Bearbeiten und zeigen Sie den XSLT Mapper an.
7. Öffnen Sie im Bereich „XML-Quelldatei“ das -Menü, wählen Sie „Öffnen“ und laden Sie Ihre Beispiel-Eingangsnachricht.
8. **XML-Query erstellen**
   a. Öffnen Sie im Bereich „XML-Zieldatei“ den Datenbank Explorer:



   b. Lassen Sie sich von dem Datenbank Explorer durch die Erstellung der XML-Query führen.
   Wenn Sie den Datenbank Explorer beendet haben, wird die XML-Query im Bereich „XML-Zieldatei“ angezeigt.
   c. Ziehen Sie das `queries`-Element nach oben auf das `xsl:template` -Element.
   d. Legen Sie fest, wie die Elemente aus der Beispiel-Eingangsnachricht auf die Query-Elemente abgebildet werden sollen.
   → Siehe *XML-Querys: Struktur und Beispiele (Workbench/ Process Engine: Systemkonnektor-Guide, Kap. 6.4, S. 63).*

Um ResultSet-Einträge besser zu Queries zuordnen zu können, fügen Sie den `query`-Elementen das Attribut `queryID`

hinzu. Im ResultSet wird dieses Attribut mit dem Wert der zugehörigen Query gefüllt.

9. Publizieren Sie den Technical Workflow.

6.4 XML-Querys: Struktur und Beispiele

Dieser Abschnitt erläutert die folgenden Themen:

6.4.1 Struktur von XML-Querys

Die grundlegende Struktur ist für alle XML-Querys identisch:

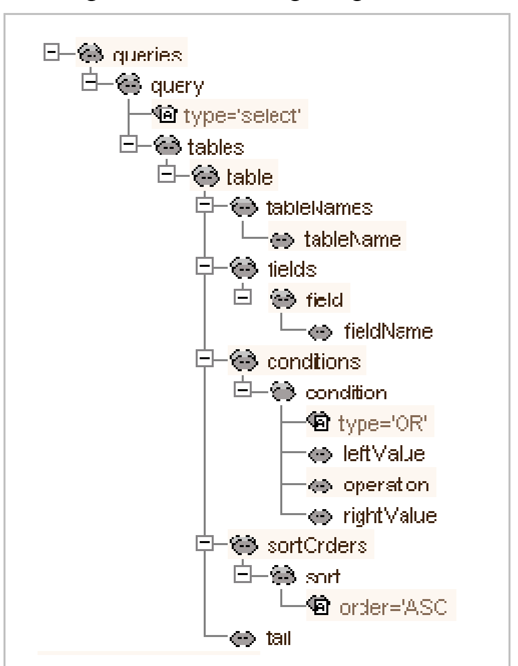
<queries>			
Umschließendes Element für ein Set von SQL-Statements.			
<query>			
Initiale Definition eines SQL-Statements. Kann mehrfach auftreten, z. B. um mehrere SQL-Anweisungen innerhalb einer Abfrage abzusenden			
Attribute	Pflichtfeld	zulässige Werte	Erläuterung

`type`	Ja	select insert update delete call	Art des SQL-Statements
`properties`	Nein	force	Mit `force` wird das SQL-Statement, das im Element `<value>` angegeben ist, direkt ausgeführt. Alle weiteren Angaben werden ignoriert.
`forceResult`	Nein	true/false	Wert „true“ ermöglicht die Rückgabe von Result-Sets aus Stored Procedures.
`<value>`			
Zur direkten Übergabe eines SQL-Statements. Dazu muss im Element `<query>` das Attribut `properties` den Wert „force“ haben. Alle weiteren Angaben werden ignoriert.			
`<tables>`			
Einleitendes XML-Element für die Definition eines SQL-Statements.			
`<table>`			
Einleitendes XML-Element für ein einzelnes SQL-Statement. Kann mit dem Attribut `type="subselect"` als Subquery in einer Hauptabfrage verwendet werden und mehrfach auftreten.			
Attribute	Pflichtfeld	zulässige Werte	Erläuterung
`Alias`	Nein	Beliebiger Bezeichner	
`Type`	Nein	subselect	Gibt an, dass es sich um eine Subquery handelt
`<tableNames>`			
Umschließendes XML-Element für alle Tabellennamen, die im SQL-Statement abgefragt werden.			
`<tableName>`			
Tabellenname, der im SQL-Statement abgefragt wird. Kann mehrfach auftreten. Entspricht dem FROM-Teil eines SQL-Statements.			
Attribute	Pflichtfeld	zulässige Werte	Erläuterung
`alias`	Nein	Beliebiger Bezeichner	Alias-Name der Tabelle
`<fields>`			
Umschließendes XML-Element für alle im SQL-Statement zu verwendenden Spaltennamen.			
`<field>`			
XML-Element für den Namen der im SQL-Statement abgefragten Tabellenspalte. Kann mehrfach auftreten. Entspricht dem Select-Teil eines SQL-Statements.			
Attribute	Pflichtfeld	zulässige Werte	Erläuterung

`Alias`	Nein	Beliebiger Bezeichner	Alias-Name der Spalte.
<fieldName>			
Name einer im SQL-Statement zu verwendenden Tabellenspalte.			
<fieldValue>			
Werte für einzelne Felder oder der Name einer Subquery. (nur INSERT, UPDATE)			
Attribute	Pflichtfeld	zulässige Werte	Erläuterung
`type`	Nein	subselect, force	subselect: Zum Ausführen einer Subquery force: Die automatische Typerkennung wird für dieses Feld ausgeschaltet. Der Text von `<fieldValue>` wird direkt an die Datenbank übergegeben. Z. B. sinnvoll, um Datenbankfunktionen wie AVG oder MAX ausführen zu lassen.
<conditions> (nicht für INSERT)			
Umschließendes XML-Element für die WHERE-Bedingung eines SQL-Statements.			
<condition>			
Werte für ein Kriterium in der WHERE-Bedingung eines SQL-Statements. Kann mehrfach auftreten.			
Attribute	Pflichtfeld	zulässige Werte	Erläuterung
`type`	Ja.	AND, OR, NOT, AND NOT, OR NOT	Wert für die Verknüpfung mit anderen Kriterien in weiteren `<condition>`-Elementen. Standard ist AND.
<leftValue>			
Spaltenname auf der linken Seite eines Kriteriums in einer WHERE-Bedingung.			
<operation>			
Operator innerhalb eines Kriteriums in einer WHERE-Bedingung, z.B. „=“, „<„, „>“, „>=“, „<=“, „IN“, „NOT IN“,…			
<rightValue>			
Enthält den Wert auf der rechten Seite eines Kriteriums in einer WHERE-Bedingung. Dabei kann es sich um einen Wert, um einen Spaltennamen oder um den Namen eines Subselect handeln.			
Attribute	Pflichtfeld	zulässige Werte	Erläuterung
`type`	Nein	subselect, force	subselect: Zum Ausführen einer Subquery force: Die automatische Typerkennung wird für dieses Feld ausgeschaltet. Der Text von `<rightValue>` wird direkt an die Datenbank übergegeben. Ist z. B. sinnvoll, um eingebaute Datenbankfunktionen AVG oder MAX auszuführen.
<sortOrders> (nur für SELECT)			

Sortiert die Ausgaben nach einem Feld auf- oder absteigend.			
<sort>			
Feld, das als Sortierkriterium verwendet wird			
Attribute	Pflichtfeld	zulässige Werte	Erläuterung
`order`	Nein.	„ASC“ - aufsteigend „DESC“ - absteigend	Sortierreihenfolge
<tail>			
Der Wert des Elements wird an die erzeugte SQL-Anfrage angehängt. Damit können Datenbank-spezifische SQL Erweiterungen hinzugefügt werden.			

Die folgende Abbildung zeigt ein SELECT als Baumstruktur:



6.4.2 Select: Daten anzeigen

SQL

```
SELECT name, address, phone, fax
  FROM addresses
  WHERE name = "Georg Miller"
    ORDER BY name ASC, address DESC
```

XML-Query

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<queries>
  <query type="select">
    <tables>
      <table>
        <tableNames>
          <tableName>addresses</tableName>
        </tableNames>
        <fields>
          <field>
            <fieldName>name</fieldName>
          </field>
          <field>
            <fieldName>address</fieldName>
          </field>
          <field>
            <fieldName>phone</fieldName>
          </field>
          <field>
            <fieldName>fax</fieldName>
          </field>
        </fields>
        <conditions>
          <condition>
            <leftValue>name</leftValue>
            <operation>=</operation>
            <rightValue>Georg Miller</rightValue>
          </condition>
        </conditions>
        <sortOrders>
          <sort>name</sort>
          <sort order="DESC">address</sort>
        </sortOrders>
      </table>
    </tables>
  </query>
</queries>
```

6.4.3 Select Distinct

Mit einem SELECT DISTINCT werden alle unterschiedlichen Werte einer Tabelle zurückgegeben.

SQL

```
SELECT DISTINCT abteilung
FROM mitarbeiter
```

XMLQuery

Um ein SELECT DISTINCT zu formulieren, haben Sie zwei Möglichkeiten:

1. DISTINCT beim Spaltennamen angeben:
```
<queries>
 <query type="select">
   <tables>
     <table>
       <tableNames>
         <tableName>Mitarbeiter</tableName>
       </tableNames>
       <fields>
         <field>
           <fieldName>DISTINCT Abteilung</
fieldName>
         </field>
       </fields>
     </table>
   </tables>
 </query>
</queries>
```
2. Mit dem Attribut `properties="force"` die Abfrage direkt übergeben:
```
<queries>
 <query type="select" properties="force">
   <value>SELECT DISTINCT Abteilung FROM
Mitarbeiter</value>
 </query>
</queries>
```

6.4.4 Daten aus verschiedenen Tabellen: Join

Ein Join ist eine Abfrage, die Datensätze aus zwei oder mehr verschiedenen Tabellenspalten zueinander in Beziehung setzt, z. B.:

SQL

```
SELECT *
  FROM Gartenmoebel, Kategorien
  WHERE Gartenmoebel.KategorieNr =
Kategorien.Kategorienummer
```

XMLQuery

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<queries>
  <query type="select">
    <tables>
      <table>
        <tableNames>
          <tableName>Gartenmoebel</tableName>
          <tableName>Kategorien</tableName>
        </tableNames>
        <fields>
          <field>
            <fieldName>*</fieldName>
          </field>
        </fields>
        <conditions>
          <condition>
            <leftValue>Gartenmoebel.KategorieNr</
leftValue>
            <operation>=</operation>
            <rightValue>Kategorien.Kategorienummer</
rightValue>
          </condition>
        </conditions>
      </table>
    </tables>
  </query>
</queries>
```

6.4.5 Datensätze einfügen: Insert

Um einer Tabelle einen Datensatz hinzuzufügen, verwenden Sie ein Insert.

SQL

```
INSERT INTO addresses (name, address, phone, fax)
VALUES ("Georg Miller", "Hauptstrasse 7", "253263",
"253264")
```

XMLQuery

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<queries>
  <query type="insert">
    <tables>
      <table>
        <tableNames>
          <tableName>addresses</tableName>
        </tableNames>
        <fields>
          <field>
            <fieldName>name</fieldName>
            <fieldValue>Georg Miller</fieldValue>
          </field>
          <field>
            <fieldName>address</fieldName>
            <fieldValue>Hauptstrasse 7</fieldValue>
          </field>
          <field>
            <fieldName>phone</fieldName>
            <fieldValue>253263</fieldValue>
          </field>
          <field>
            <fieldName>fax</fieldName>
            <fieldValue>253264</fieldValue>
          </field>
        </fields>
      </table>
    </tables>
  </query>
</queries>
```

6.4.6 Datensätze aktualisieren: Update

Um bestehende Datensätze in einer Tabelle zu aktualisieren, verwenden Sie das Update.

SQL

```
UPDATE addresses
SET address = "Hauptstrasse 8"
WHERE name = "Georg Miller"
```

XMLQuery

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<queries>
  <query type="update">
    <tables>
      <table>
        <tableNames>
          <tableName>addresses</tableName>
        </tableNames>
        <fields>
          <field>
            <fieldName>address</fieldName>
            <fieldValue>Hauptstrasse 8</fieldValue>
          </field>
        </fields>
        <conditions>
          <condition>
            <leftValue>name</leftValue>
            <operation>=</operation>
            <rightValue>Georg Miller</rightValue>
          </condition>
        </conditions>
      </table>
    </tables>
  </query>
</queries>
```

6.4.7 UpdateOrInsert

Dieses SQL-Statement bildet die SQL-Statements Insert und Update ab. Damit wird versucht, einen Datensatz zu aktualisieren (Update); falls dies nicht möglich ist, weil der Datensatz noch nicht vorhanden ist, wird der Datensatz eingefügt (Insert). Das Update wird zuerst ausgeführt, weil es weniger zeitintensiv ist als das Insert.

```
<queries>
  <query type="updateOrInsert">
    <tables>
      <table>
        <tableNames>
          <tableName>OrderTable</tableName>
        </tableNames>
        <fields>
          <field skip="update">
            <fieldName>OrderNr</fieldName>
            <fieldValue>Kunde11</fieldValue>
          </field>
          <field>
            <fieldName>Kunde</fieldName>
            <fieldValue>123</fieldValue>
          </field>
          <field>
            <fieldName>Text1</fieldName>
            <fieldValue>info1</fieldValue>
          </field>
          <field>
            <fieldName>Text2</fieldName>
            <fieldValue>info2</fieldValue>
          </field>
        </fields>
        <conditions>
          <condition type="OR">
            <leftValue>OrderNr</leftValue>
            <operation>=</operation>
            <rightValue>Kunde11</rightValue>
          </condition>
        </conditions>
      </table>
    </tables>
  </query>
</queries>
```

6.4.8 Datensätze löschen: Delete

Zum Löschen von Datensätzen verwenden Sie Delete. Im folgenden Beispiel werden alle Daten von Georg Miller gelöscht.

SQL

```
DELETE
FROM addresses
WHERE name="Georg Miller"
```

XMLQuery

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<queries>
  <query type="delete">
    <tables>
      <table>
        <tableNames>
          <tableName>addresses</tableName>
        </tableNames>
        <conditions>
          <condition>
            <leftValue>name</leftValue>
              <operation>=</operation>
              <rightValue>Georg Miller</rightValue>
          </condition>
        </conditions>
      </table>
    </tables>
    </query>
</queries>
```

6.4.9 Mehrere SQL-Statements in einer Abfrage

XML-Query

Dynamische XML-Querys können mehrere Statements enthalten:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<queries>
  <query type="insert">
    <tables>
      <table>
        <tableNames>
```

```
          <tableName>eins</tableName>
        </tableNames>
        <fields>
          <field>
            <fieldName>id</fieldName>
            <fieldValue>1</fieldValue>
          </field>
          <field>
            <fieldName>meldung</fieldName>
            <fieldValue>
              spießbraten köcheln
            </fieldValue>
          </field>
          <field>
            <fieldName>datum</fieldName>
            <fieldValue>
              2003-03-20 10:20:00.000000000
            </fieldValue>
          </field>
          <field>
            <fieldName>wert</fieldName>
            <fieldValue>10.12</fieldValue>
          </field>
        </fields>
      </table>
    </tables>
  </query>
  <query type="select">
  <tables>
    <table>
      <tableNames>
        <tableName>eins</tableName>
      </tableNames>
      <fields>
        <field>
          <fieldName>*</fieldName>
        </field>
      </fields>
      <conditions>
        <condition type="OR">
          <leftValue>id</leftValue>
          <operation>=</operation>
          <rightValue>1</rightValue>
          </condition>
```

```
        </conditions>
      </table>
    </tables>
  </query>
  <query type="update">
    <tables>
      <table>
        <tableNames>
          <tableName>eins</tableName>
        </tableNames>
        <fields>
          <field>
            <fieldName>id</fieldName>
            <fieldValue>2</fieldValue>
          </field>
        </fields>
        <conditions>
          <condition type="OR">
            <leftValue>id</leftValue>
              <operation>=</operation>
            <rightValue>1</rightValue>
          </condition>
        </conditions>
      </table>
    </tables>
  </query>
  <query type="delete">
    <tables>
      <table>
        <tableNames>
          <tableName>eins</tableName>
        </tableNames>
      </table>
    </tables>
  </query>
</queries>
```

6.4.10 SQL-Statements direkt übergeben: Force

XMLQuery

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/
XSL/Transform"
                version="1.0">
  <xsl:output method="xml" encoding="ISO-8859-1"/>
    <xsl:template match="/">
      <queries>
        <query type="select" properties="force">
          <value>
create table trade_user ( id_user NUMBER(20,0)
PRIMARY KEY, since_user NUMBER(15,0), lastlogin_user
NUMBER(15,0), sex_user NUMBER(1,0), language_user
VARCHAR2(5), title_user VARCHAR2(4000), firstname_
user VARCHAR2(200), lastname_user VARCHAR2(200),
company_user NUMBER(20,0), function_user
VARCHAR2(4000), street_user VARCHAR2(4000), zip_
user VARCHAR2(4000), land_user VARCHAR2(4000), city_
user VARCHAR2(4000), phone_user VARCHAR2(4000),
mobile_user VARCHAR2(4000), fax_user
VARCHAR2(4000), email_user VARCHAR2(200) NOT NULL
UNIQUE, web_user VARCHAR2(4000), via_user
VARCHAR2(4000), enable_user NUMBER(2,0), comment_
user VARCHAR2(4000), abbo_news_user NUMBER(1,0),
abbo_mail_user NUMBER(1,0), keyaccount_user
NUMBER(20,0), lastforum_user NUMBER(20,0),
contentarea_user VARCHAR2(4000), deliveryplace_user
VARCHAR2(4000) DEFAULT '1', originplace_user
VARCHAR2(4000) DEFAULT '1', sms_mail_user
VARCHAR2(1900), customer_protection_user
NUMBER(1,0) )
          </value>
        </query>
        <query type="select" properties="force">
          <value>
insert into xtrade_user( id_user, firstname_user,
lastname_user, company_user, email_user ) values
(1,'Hans','Müller',1,'h.mueller@karstadt.de')
          </value>
        </query>
        <query properties="force">
          <value>DROP TABLE "ORATEST"."TESTTABLE"</
value>
        </query>
      </queries>
```

```
    </xsl:template>
</xsl:stylesheet>
```

6.4.11 Subqueries: SubSelects

Sie können SQL-Abfragen verschachteln, um das Ergebnis eines SubSelects in demselben Select auszuwerten.

SQL

```
SELECT LieferscheinId , ErstellungsDatum
  FROM LieferscheinTable
  WHERE
    factory=(SELECT factory FROM SpeditionsTable
      WHERE status!=X AND TransportId=10000037)
    AND gate=(SELECT gate FROM SpeditionsTable
      WHERE status!=X AND TransportId=10000037 )
    AND status = F
```

XMLQuery

Für jedes SubSelect wird ein zusätzliches `table`-Element mit den Attributen `type="subselect"` und `alias=[NameDesSubselects]` erstellt. Dieses `table`-Element muss vor dem Haupt-Select stehen.

Das Ergebnis des SubSelects wird in das Haupt-Select in `field` und/oder `condition`-Elementen über das `Attribut type="subselect"` eingebunden, dabei ist der Wert des `field` und/oder `condition`-Elements dann der Name des SubSelects:

```
[…]
<tables>
  <table type="subselect" alias="subFactory">
    <tableNames>
      <tableName>SpeditionsTable</tableName>
    </tableNames>
    <fields>
      <field>
        <fieldName>factory</fieldName>
      </field>
    </fields>
    <conditions>
      <condition type="AND">
        <leftValue>status</leftValue>
        <operation>!=</operation>
```

```
        <rightValue>X</rightValue>
      </condition>
      <condition type="AND">
        <leftValue>TransportId</leftValue>
        <operation>=</operation>
        <rightValue>
          <xsl:value-of select="FormPrepare/
            Response/Panel/Transporte/SLB-Nr"/>
        </rightValue>
      </condition>
    </conditions>
  </table>
  <table type="subselect" alias="subGate">
    <tableNames>
      <tableName>SpeditionsTable</tableName>
    </tableNames>
    <fields>
      <field>
        <fieldName>gate</fieldName>
      </field>
    </fields>
    <conditions>
      <condition type="AND">
        <leftValue>status</leftValue>
        <operation>!=</operation>
        <rightValue>X</rightValue>
      </condition>
      <condition type="AND">
        <leftValue>TransportId</leftValue>
        <operation>=</operation>
        <rightValue>
          <xsl:value-of select="FormPrepare/
          Response/Panel/Transporte/SLB-Nr"/>
        </rightValue>
      </condition>
    </conditions>
  </table>
  <table>
    <tableNames>
      <tableName>LieferscheinTable</tableName>
    </tableNames>
    <fields>
      <field>
        <fieldName>LieferscheinId</fieldName>
```

```
        </field>
        <field>
          <fieldName>ErstellungsDatum</fieldName>
        </field>
      </fields>
      <conditions>
        <condition type="AND">
          <leftValue>factory</leftValue>
          <operation>=</operation>
          <rightValue type="subselect">
            subFactory
          </rightValue>
        </condition>
        <condition type="AND">
          <leftValue>gate</leftValue>
          <operation>=</operation>
          <rightValue type="subselect">
            subGate
          </rightValue>
        </condition>
        <condition type="AND">
          <leftValue>status</leftValue>
          <operation>=</operation>
          <rightValue>F</rightValue>
        </condition>
      </conditions>
    </table>
  </tables>
```

6.4.12 Stored Procedures aufrufen: Call

In einer Stored Procedure können ganze Abläufe von Anweisungen unter einem Namen gespeichert, auf dem Datenbankserver zur Verfügung gestellt und ausgeführt werden.

Das folgende Beispiel illustriert den Aufruf von Stored Procedures mit Hilfe des Elements `<method name="[NAME]">` mit und ohne Parameterübergabe:

XML-SQL

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/
XSL/Transform" version="1.0">
```

```
<xsl:output method="xml" encoding="ISO-8859-1"/>
  <xsl:template match="/">
    <queries>
      <query type="call">
        <method name="TESTPROC_WITHOUTPARAM"/>
      </query>
      <query type="call">
        <method name="TESTPROC_IN">
          <parameter direction="IN"
type="DECIMAL">2</parameter>
        </method>
      </query>
      <query type="call">
        <method name="TESTPROC_OUT">
          <parameter direction="OUT" type="VARCHAR"/>
        </method>
      </query>
      <query type="call">
        <method name="TESTPROC_INOUT">
          <parameter direction="INOUT"
type="NUMERIC">4</parameter>
        </method>
      </query>
      <query type="call">
        <method name="TESTPROC_INOUT2">
        <parameter direction="IN" type="NUMERIC">5</
parameter>
        <parameter direction="OUT"
type="NUMERIC">5</parameter>
        </method>
      </query>
      <query type="call">
        <method name="TESTFUNC_OUT">
          <result type="DATE"/>
        </method>
      </query>
      <query type="call">
        <method name="TESTFUNC_INOUT">
        <parameter type="NUMERIC" direction="IN">2</
parameter>
          <result type="NUMERIC"/>
        </method>
      </query>
    </queries>
```

```
  </xsl:template>
</xsl:stylesheet>
```

6.5 Dialogbeschreibungen

Dieser Abschnitt erläutert die folgenden Themen:

- *Dialog „Datenbankverbindungen und Query-Typ“, S. 81*
- *Dialog „Metadaten“, S. 84*
- *Dialog „Statische SQL-Query“, S. 85*
- *Dialog „Ergebnis der Query“, S. 85*
- *Dialog „Datenbank Connection Pooling“, S. 87*

6.5.1 Dialog „Datenbankverbindungen und Query-Typ“

Dieser Dialog bietet folgende Optionen:

Datenbankverbindung

- **Voreinstellung**

  Zur Auswahl einer vorkonfigurierten Datenbank, um die Felder „Datenbank-URL“ und „Datenbanktreiber-Klasse“ sinnvoll zu belegen.

  Bei Auswahl von „IS Log Database“ verbindet sich der Database Connector mit der Logging/Monitoring-Datenbank der inubit Suite 6. Für die Verbindung wird die Konfiguration der inubit Process Engine verwendet, deswegen werden alle anderen Felder in diesem Bereich deaktiviert.

  Falls Ihre Datenbank nicht vorhanden ist, wählen Sie die Voreinstellung „Custom“ und füllen Sie die Felder manuell aus.

  Unter Windows können Sie auf die JDBC-ODBC-Bridge ausweichen.

Wählen Sie die JDBC-ODBC-Bridge nur, wenn Sie keinen JDBC-Zugriff auf Ihrer Datenbank einrichten können. Eine JDBC-ODBC-Bridge hat eine schlechtere Performance, der Funktionsumfang kann eingeschränkt sein und die automatische Typerkennung ist deutlich langsamer.

- **Datenbank-URL**

Wird vorbelegt, wenn Sie eine vorkonfigurierte Datenbank gewählt haben.

Geben Sie den Host, den Port und den Datenbanknamen an.

- **Datenbanktreiber-Klasse**

  Entspricht der JDBC-Verbindung der eingesetzten Datenbank. Um eine andere Datenbank zu verwenden, müssen Sie einen passenden Treiber installieren.

  → Siehe *Treiber installieren (Process Engine: Administrator- und Entwickler-Guide, Kap. 3.4, S. 51).*

Nach Installation eines JDBC-Treiber für den MS SQL Server müssen Sie die inubit Process Engine neu starten.

Um eine Adabas Datenbank einzusetzen, kopieren Sie die Treiberklasse direkt in das Verzeichnis `<iS-installdir>/server/JBoss/server/default/lib` bzw. `<iS-installdir>/server/webapps/ibis/WEB-INF/lib`.

- **Benutzer/Passwort**

  Eingabe ist abhängig von Ihrer Datenbank.

- **Spezielles Kodierung verwenden**

  Wenn Sie keinen anderen Zeichensatz angeben, wird standardmäßig UTF8 benutzt.

- **Connection Pooling**

  Aktiviert das Connection Pooling.

  Wenn das Connection Pooling aktiviert ist, werden physische Verbindungen zu einer Datenbank wiederverwendet. Dies beschleunigt die Abarbeitung der Abfragen. Die Verbindungen werden nach ihrer Verwendung nicht abgebaut, sondern gespeichert und sind für nachfolgende Zugriffe verfügbar. So muss nicht bei jedem Zugriff auf die Datenbank erneut zeitintensiv eine Verbindung aufgebaut werden.

  Der Button „Einstellungen“ öffnet den *Dialog „Datenbank Connection Pooling“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 6.5.5, S. 87)* zur Konfiguration des Connection Poolings.

- **Spezielle Verbindungsparameter**

  Zum Erstellen von Parametern wie z. B. Timeouts, Anzahl offener Verbindungen etc., die genutzt werden, um die Datenbankverbindung abhängig von Ihrer verwendeten Datenbank zu konfigurieren. Die Parameter definieren Sie als Name-Wert-Paare.

Verbindungstest

- **Verbindung testen**

  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

Funktion

Mit der Auswahl legen Sie fest, welche Aufgabe der Database Connector ausführen soll:

- **Eingangsnachricht als Query ausführen**

  Der Database Connector führt Abfragen aus Eingangsnachrichten aus.

  Wenn möglich, werden sogenannte „prepared statements“ erzeugt. Wenn die Eingangsnachricht mehrere Abfragen enthält, dann wird überprüft, ob diese denselben Typ haben und auf dieselben Tabellenspalten zugreifen. Wenn ja, dann wird ein Befehl erzeugt, der mehrfach ausgeführt wird. Dieses Vorgehen erhöht die Performance.

  → Siehe *Dynamische XML-Query erstellen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 6.3, S. 61).*

- **Statische Query ausführen**

  Der Database Connector führt eine fest definierte Abfrage aus. Die Abfrage geben Sie im *Dialog „Statische SQL-Query“ (Workbench/ Process Engine: Systemkonnektor-Guide, Kap. 6.5.3, S. 85)* ein.

  → Siehe *Statische SQL-Query erstellen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 6.2, S. 60).*

- **Metadaten lesen**

  Der Database Connector fragt die Struktur der Datenbank und weitere Informationen ab. Diese Infos benötigen Sie z. B., wenn Sie in einer Abfrage Tabellen- und Spaltennamen abbilden.

  → Siehe *Metadaten abfragen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 6.1, S. 60).*

Einstellungen

- **Automatische Typenerkennung der Datenbankobjekte**

  Wenn markiert, dann werden Datentypen automatisch erkannt. Wenn z. B. mit der Query ein Datum in die Datenbank eingefügt werden soll, dann wird das Datum korrekt mit dem Typ „Date“ übergeben.

- **Typerkennung cachen**

  Aktivieren Sie die Checkbox, um die Tabellendefinitionen vorzuhalten, damit sie nicht bei jeder Ausführung des Database Connectors von der Datenbank abgefragt werden müssen.

- **Auto Commit (keine Transaktion)**

  Bei einer Ausführung des Database Connectors können mehrere SQL-Statements ausgeführt werden. Mit dieser Option bestimmten Sie, wann die Änderungen dauerhaft in der Datenbank gespeichert werden:

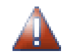
- Wenn das Auto-Commit aktiviert ist, dann wird für jede Änderung automatisch ein Commit oder Rollback (im Fehlerfall) durchgeführt.
- Wenn die Option deaktiviert ist, dann werden alle Änderungen zusammen in einer Transaktion durchgeführt. Falls bei dieser Transaktion ein Fehler auftritt, dann wird keine weitere Änderung ausgeführt und alle vorher bereits ausgeführten Änderungen werden rückgängig gemacht.
- **Fehler überspringen**
  Wenn aktiviert, dann werden Datenbankanfragen auch ausgeführt, nachdem ein Fehler aufgetreten ist.

■ **Werte aus der Query trimmen**
Bei Werten in Feldern und Bedingungen werden alle vorlaufenden und nachfolgenden Leerzeichen gelöscht. Leerzeichen innerhalb eines Wertes oder einer Bedingung werden nicht bearbeitet.

6.5.2 Dialog „Metadaten“

In diesem Dialog legen Sie fest, welche Informationen über die Datenbank ausgegeben werden sollen.

Diese Metadaten werden als XML-Nachricht ausgegeben, dabei wird für jedes Metadatum ein XML-Element erzeugt. Sie können die Namen dieser Elemente und des Wurzelelements angeben.



Die Namen müssen folgenden Regeln entsprechen:
- Namen können Buchstaben, Ziffern und andere Zeichen enthalten.
- Namen dürfen nicht mit einer Zahl oder einem Satzzeichen beginnen.
- Namen dürfen nicht mit xml, XML, Xml etc. beginnen).
- Namen dürfen keine Leerzeichen enthalten.

Wenn ein Name unzulässig ist, dann erhält das Element den Namen „InvalidXmlTag“. Die Metadaten werden als Wert des `name`-Attributs ausgegeben.

Mit einem Klick auf den Button „Vorschau erzeugen“ erhalten Sie eine Vorschau auf das Ausgabeformat der Metadaten.

■ **Nur diese Tabelle**
Standardmäßig werden die Daten aller Tabellen im Schema ausgelesen. Mit dieser Option könne Sie das Auslesen der Metadaten auf eine Tabelle beschränken.

■ **Root-Element**
XML-Wurzelelement.

■ **Datentyp**
Liest den Datentyp der Tabellenspalten in der Datenbank aus.

- **Typ-Bezeichner**
  Liest die Tabellenspaltennamen aus.
- **Spaltenbreite**
  Maximale Länge eines Feld in der Tabellenspalte.
- **Dezimalstellen**
  Zeigt an, wie viele Stellen hinter dem Komma Werte maximal haben dürfen, die in einer Spalte mit Gleitkommazahlen enthalten ist.
- **NULL-Werte**
  Liest aus, welche Tabellenspalten leere Einträge haben dürfen.
- **Bemerkungen**
  Liest aus, zu welchen Tabellenspalten Kommentare existieren.

6.5.3 Dialog „Statische SQL-Query“

In diesem Dialog definieren Sie die statische Query. Die Abfrage darf nur ein SQL-Statement enthalten.

6.5.4 Dialog „Ergebnis der Query“

In diesem Dialog legen Sie fest, wie das Ergebnis der Query formatiert werden soll. Mit Hilfe der Vorschau können Sie die Formatierung überprüfen.

Format des Ergebnisses

- **Textwerte aus der Datenbank trimmen**
  Bei Textfeldern werden vorlaufende und nachfolgende Leerzeichen gelöscht. Leerzeichen, die nicht am Anfang oder am Ende eines Textfeldes stehen, werden ignoriert.
- **NULL-Werte durch Attribut kennzeichnen**
  Damit ist es möglich, in der Ausgabe-Nachricht zwischen einem leeren Wert und dem Wert NULL zu unterscheiden. Der Wert NULL wird mit dem Attribut `Value null="true"` gekennzeichnet.
- **Ergebnis immer mit „ResultSets“-Tag umhüllen**
  Wenn markiert, dann wird z. B. als Ergebnis eines einfachen Inserts eines Datensatzes

```
<ResultSets>
 <rows type="insert">1</rows>
```

```
</ResultSets>
```

ausgegeben, statt nur

```
<rows type="insert">1</rows>
```

- **Datums- und Zeitfelder im XML-Schema Format ausgeben**
  Wenn markiert, dann werden Zeit- und Datumsstempel im entsprechenden Format einschließlich der Angabe von Zeitzonen mit ausgegeben, z. B. 2009-10-19T12:25:09.00+02.
  Ansonsten wird die Datums- und Zeitangabe ohne Zeitzone in der lokalen Zeit ausgegeben.

Bezeichner der XML-Elemente

In den Textfeldern bestimmen Sie die Namen der Elemente, die in der XML-Ausgangsnachricht angelegt werden.

- **Formatierungstyp**
  - **Unkomprimiert**
    Die Ergebnisse werden durch Elemente gekapselt. Sie können die Elementbezeichner manuell ändern. Beispiel:

```
<ResultSet>
 <Row>
   <Column id="1">
     <Name>Street</Name>
     <Value>Unter den Linden</Value>
   </Column>
 </Row>
```

  - **Komprimiert**
    Die Ergebnisse werden durch Elemente gekapselt (außer Feldnamen und -werte). Sie können die Elementbezeichner ändern. Beispiel:

```
<ResultSet>
 <R>
   <C>Street</C>
 </R>
 <R>
   <C>Unter den Linden</C>
 </R>
```

  - **Namen als Tags**
    Die Namen der Spalten werden als Element-Namen verwendet. Falls die Spaltennamen nicht den Konventionen von XML-Elementen entsprechen, werden die Elemente mit „InvalidXmlTag“ angelegt und der Spaltenname wird als Wert des Attributs „Name“ ausgegeben.
  - **Benutzerdefiniert**
    Alternativ können Sie zur Ausgabe eine Java-Klasse nutzen, welche das SQL-Ergebnis nach XML umformt. Geben Sie den kompletten Namen ein.

Die Java-Klasse muss das Interface `com.inubit.ibis.utils.query.ResultSetOutput` (im SDK enthalten) implementieren.

- **Zeilennummer**
  Wenn markiert, dann wird bei den Elementen ROW und COLUMN das Attribut ID mit dem Wert der jeweiligen laufenden Nummer ausgegeben, z. B.:

```
<ResultSet>
 <Row id="1">
   <Column id="1">Name</Column>
   <Column id="2">Street</Column>
 </Row>
```

Vorschau

Mit einem Klick auf den Button „Vorschau erzeugen“ erhalten Sie eine Vorschau auf das Ausgabeformat der Query.

6.5.5 Dialog „Datenbank Connection Pooling“

In diesem Dialog konfigurieren Sie das Connection Pooling.

Wenn das Connection Pooling aktiviert ist, werden physische Verbindungen zu einer Datenbank wiederverwendet. Dies beschleunigt die Abarbeitung der Abfragen. Die Verbindungen werden nach ihrer Verwendung nicht abgebaut, sondern gespeichert und sind für nachfolgende Zugriffe verfügbar. So muss nicht bei jedem Zugriff auf die Datenbank erneut zeitintensiv eine Verbindung aufgebaut werden.

Wiederverwendung des Connection Pools:
Alle Database Connectoren, bei denen die Werte der Einstellungen „URL“, „User“ und „AutoCommit“ übereinstimmen, verwenden denselben Connection Pool.

Verbindungspooling

- **Max. Anzahl aktiver Verbindungen im Pool**
  Legt fest, wie viele Verbindungen gleichzeitig aktiv sein dürfen.
  Eine zu kleine Anzahl von Verbindungen im Connection Pool kann zu Wartezeiten führen; eine zu große Zahl kann das Datenbanksystem unnötig belasten, weil alle Verbindungen einen gewissen Verwaltungsaufwand seitens des Datenbanksystems erfordern.
- **Max. Anzahl inaktiver Verbindungen im Pool**

Legt fest, wie viele Verbindungen im Connection Pool gleichzeitig inaktiv sein dürfen.

- **Min. Anzahl inaktiver Verbindungen im Pool**
  Legt fest, wie viele inaktive Verbindungen der Connection Pool immer hält.
- **Max. Zeit des Pools, um auf eine Verbindung zu warten**
  Legen Sie fest, wie lange der Pool höchstens warten soll, bis der Connector eine Connection vom Pool erhält.
- **Aktion, wenn Pool voll ist**
  Wählen Sie aus, welche Aktion ausgeführt werden soll, wenn alle Verbindungen im Connection Pool in Benutzung sind.
  - **Fehlschlagen mit Exception**
    Das Anfordern einer Verbindung aus dem Connection Pool wird abgewiesen, damit scheitert die Datenbankanfrage und eine Fehlermeldung wird erzeugt.
  - **Pool blockieren**
    Das Anfordern einer Verbindung wird blockiert.
  - **Neue Verbindung erzeugen**
    Eine weitere Verbindung wird erzeugt.

Validierung

Der Pool nutzt eine Verbindung erst nach einer erfolgreichen Prüfung. Falls die Prüfung fehlschlägt, wird eine neue Verbindung erzeugt. Damit werden Fehler am Datenbank Connector vermieden und die Stabilität bei hoch ausgelasteten Datenbanken wird verbessert.

- **Test beim Entnehmen von Verbindungen aus dem Pool**
  Testet eine Datenbankverbindung des Pools auf Funktionalität, wenn diese aus dem Connection Pool genommen wird.
- **Test beim Zurücklegen von Verbindungen in den Pool**
  Testet eine Datenbankverbindung des Pools auf Funktionalität, wenn diese nach der Benutzung zurück gelegt wird.
- **Test, während die Verbindungen inaktiv sind**
  Testet eine Datenbankverbindung des Pools auf Funktionalität, während sie nicht verwendet wird.
- **Validierung SQL-Query**
  Prüft und validiert die Verbindung mit einer SQL-Query. Mit starten Sie den Test.
- **Zeit zwischen Prüfung inaktiver Verbindungen**
  Zeit zwischen zwei Prüfungen inaktiver Verbindungen. Angabe in Millisekunden.
- **Anzahl zu prüfender, inaktiver Verbindungen**

Überprüft inaktive Verbindungen in regelmäßigen Zeitabständen darauf, ob diese noch die korrekte Verbindungsfunktionalität aufweisen. Scheitert dieser Test, wird die Verbindung aus dem Connection Pool entfernt.

- **Min. Zeit für inaktive Verbindungen im Pool**
  Zeit, in der eine ungenutzte Verbindung im Pool offen bleibt, bis sie geschlossen wird.
- **Setze Vorgabewerte (Button)**
  Stellt die ausgelieferten Standardwerte wieder her.

7 Database Object Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Der Database Object Connector (DBO Connector) ermöglicht den direkten Zugriff auf Geschäftsobjekte, deren Daten in hierarchisch strukturierten Datenbanktabellen gespeichert sind.

Konnektortypen

Abhängig von seiner Konfiguration hat ein DBO Connector folgende Funktionen:

- **Input Connector**
  - Als Event-Listener überwacht und archiviert ein DBO Connector Änderungen an Geschäftsobjekten.

    → Siehe *Events mit Event-Listener überwachen und archivieren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 7.2, S. 94).*
  - Als Archiv-Prozessor schützt ein DBO Input Connector ein Event-Archiv vor dem Überlauf.

    → Siehe *Event-Archiv mit Archiv-Prozessor überwachen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 7.3, S. 97).*
- **Medium Connector**

  Zum Erstellen, Lesen, Bearbeiten oder Löschen eines oder mehrerer Geschäftsobjekte.

  → Siehe *Geschäftsobjekte lesen, erstellen, bearbeiten oder löschen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 7.1, S. 92).*

Unterstützte Datenbanken

Der DBO Connector unterstützt folgende Datenbanken:

- Oracle
- MySQL 5.0, 5.1 und 6
- MS-SQL

7.1 Geschäftsobjekte lesen, erstellen, bearbeiten oder löschen

Für den Zugriff auf Geschäftsobjekte nutzen Sie einen DBO Medium Connector. Mit der Eingangsnachricht steuern Sie, welche Operation der Konnektor ausführen soll, steuern Sie über den Inhalt der Eingangsnachrichten des Konnektors.

Die Kommunikation mit der Datenbank erfolgt grundsätzlich transaktional: es wird immer die gesamte Operation durchgeführt. Falls ein Fehler auftritt, dann wird die gesamte Operation zurückgerollt.

Nach jeder erfolgreich durchgeführten Transaktion gibt der Konnektor die geänderten Geschäftsobjekte aus. Nach einem `Delete` wird eine Information ausgegeben, welche Geschäftsobjekte gelöscht wurden.

Mapping Template

Alle Eingangsnachrichten müssen eine bestimmte Struktur haben. Diese Struktur ist in den mitgelieferten Templates beschrieben. Es gibt für jede Operation ein eigenes Template: `Insert` (mit und ohne Primärschlüssel), `Select`, `Update` und `Delete`.

Sie finden alle Templates im Repository im Verzeichnis `Global/System/Mapping Template/Database Object Connector`.

So gehen Sie vor

1. Erstellen Sie einen DBO Medium Connector.
2. Konfigurieren Sie im *Dialog „Datenbankeinstellungen“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 7.6.1, S. 102)* die Datenbankverbindung und testen Sie diese.
3. Geben Sie im Feld „Plugin-ID“ eine ID ein. Die ID muss innerhalb Ihrer inubit Suite 6 je Datenbank eindeutig sein.

   Die ID wird benötigt, wenn Sie zusätzlich einen DBO Input Connector als Event-Listener einrichten: Der Event-Listener nutzt die ID, um den DBO Medium Connector zu identifizieren, der die zu überwachenden Events erzeugt.

   → Siehe *Events mit Event-Listener überwachen und archivieren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 7.2, S. 94).*
4. Klicken Sie auf „Weiter“.

   Der *Dialog „DBO-Konfigurationsdatei“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 7.6.3, S. 105)* öffnet sich und zeigt das zu Ihrer Datenbank passende Template für die Konfigurationsdatei an.
5. **Konfigurationstemplate anpassen**:

   In der angezeigten Konfigurationsdatei bilden Sie die Struktur Ihrer Datenbanktabellen auf die vorgegebene XML-Struktur ab.

   → Siehe *Konfigurationsdatei: Feldbeschreibungen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 7.4, S. 97).*

a. Geschäftsobjekte definieren:

   Geschäftsobjekte werden im `DBStructureDescription`-Element beschrieben. Sie können beliebig viele `DBStructureDescription`-Elemente konfigurieren und beliebig tiefe Hierarchien durch Untertabellen abbilden.

   Kopieren Sie das Element `FATHER_CHILD-DBStructureDescription` und passen Sie dieses an Ihre Tabellenstruktur an.

   Das Beispielelement `DBStructureDescription` mit dem Attribut `DBStructureName=FATHER_CHILD` zeigt, wie Constraints für Primär- und Fremdschlüssel innerhalb `DBStructureDescription` abgebildet werden.

b. Optional: Datumsformat für JDBC-Session anpassen

   - **MySQL**: Das Datumsformat `xs:dateTime` ist fest konfiguriert, so dass es nicht einmalig für die gesamte Session gesetzt werden kann.
   - **Oracle**: Die Konfigurationsdatei initialisiert die JDBC-Session mit dem Datumsformat `xs:dateTime (yyyy-mm-dd"T"hh24:mi:ss)`.

     → Siehe *Konfigurationsdatei: Feldbeschreibungen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 7.4, S. 97).*

c. Optional:

   Sie können die angepasste Konfigurationsdatei ins Repository oder ins Dateisystem speichern, um die Konfigurationsdatei in anderen DBO-Konnektoren wiederzuverwenden.

6. Klicken Sie auf „Fertig stellen“. Der Assistent schließt sich.

7. **Datenbankprozedur `getNewID()` erstellen**

   Die Prozedur wird beim Einfügen neuer Geschäftsobjekte benötigt, um jeweils eine eindeutige Geschäftsobjekt-ID zu ermitteln.

   → Siehe *Datenbankprozedur getNewID erstellen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 7.5, S. 99).*

8. **Eingangsnachricht erstellen**

   a. Erstellen Sie einen XSLT Converter.

   b. Um ein Template als Zielstruktur zu laden, öffnen Sie im Bereich „XML-Zieldatei“ das -Menü und wählen „Öffnen von > Repository“. Der Repository Browser öffnet sich.

   c. Navigieren Sie in das Verzeichnis `/Global/System/Mapping Template/Database Object Connector`.

   d. Wählen Sie ein Template aus und klicken Sie auf „OK“, um den Repository Browser zu schließen.

   e. Passen Sie das Template an die Strukturen Ihres Geschäftsobjekts an.

   f. Verbinden Sie den XSLT Converter mit dem DBO Connector.

7.2 Events mit Event-Listener überwachen und archivieren

Als Event-Listener überwacht ein DBO Input Connector Änderungen an Geschäftsobjekten.

Dazu überprüft der DBO Input Connector in regelmäßigen Abständen, ob in einer Eventtabelle eine neue Eventbenachrichtigung steht. Wenn dies der Fall ist, dann liest der DBO Input Connector das Geschäftsobjekt ein, welches zu dem Event gehört, und gibt dieses Geschäftsobjekt aus. Wenn das Event ein „Delete“ war, wird nur der Primärschlüssel ausgegeben. Das verarbeitete Event wird in die Event-Archiv-Tabelle verschoben.

Prinzipielles Vorgehen

1. **Eventtabelle einrichten**
   In diese Tabelle werden alle Ereignisse (Insert, Delete, Update, Select) geschrieben, die in einer überwachten Datenbanktabelle auftreten.
2. **Trigger setzen**
   Die SQL-Trigger müssen auf alle Tabellen gesetzt werden, in denen Events überwacht werden sollen. Sobald in einer überwachten Tabelle z. B. ein Insert passiert, löst der Trigger aus und schreibt das Event in die Eventtabelle.
3. **DBO Input Connector erstellen**
   Der DBO Input Connector liest das Event aus der Eventtabelle, übergibt das betroffene Geschäftsobjekt dem Workflow und stößt damit die Workflowausführung an.
4. **Optional: Tabelle zur Archivierung einrichten**
   Bereits verarbeitete Events können in einer Archiv-Tabelle gespeichert werden.

So gehen Sie vor

1. **Eventtabelle erstellen**
   - **Für Oracle** mit Sequence:

```
CREATE TABLE  IS_EVENTS (
ID NUMBER (10, 0) DEFAULT 0 NOT NULL,
IS_PLUGIN_ID VARCHAR2(10) DEFAULT '' NOT NULL,
PK1 VARCHAR2(100) DEFAULT '' NOT NULL,
PK2 VARCHAR2(100) DEFAULT '' NULL,
PK3 VARCHAR2(50) DEFAULT '' NULL,
PK4 VARCHAR2(50) DEFAULT '' NULL,
PK5 VARCHAR2(50) DEFAULT '' NULL,
PK6 VARCHAR2(50) DEFAULT '' NULL,
PK7 VARCHAR2(50) DEFAULT '' NULL,
PK8 VARCHAR2(50) DEFAULT '' NULL,
PK9 VARCHAR2(50) DEFAULT '' NULL,
```

```
IS_STRUCTURE_NAME VARCHAR2(50) DEFAULT '' NOT
NULL,
VERB VARCHAR2(10) DEFAULT '' NOT NULL,
PRIORITY NUMBER(10, 0) DEFAULT 0 NOT NULL,
TIMESTAMP DATE DEFAULT SYSDATE NOT NULL,
STATUS NUMBER (10, 0) DEFAULT 0 NOT NULL,
RETRIEVE_TIME DATE DEFAULT SYSDATE NOT NULL
)
CREATE SEQUENCE SEQ_IS_EVENTS INCREMENT BY 1
START WITH 1 NOMAXVALUE   NOCYCLE CACHE 10
```

- **Für MySQL**:

```
CREATE TABLE  IS_EVENTS (
ID NUMERIC (10, 0) DEFAULT 0 NOT NULL,
IS_PLUGIN_ID VARCHAR(10) DEFAULT '' NOT NULL,
PK1 VARCHAR(100) NOT NULL,
PK2 VARCHAR(100),
PK3 VARCHAR(50),
PK4 VARCHAR(50),
PK5 VARCHAR(50),
PK6 VARCHAR(50),
PK7 VARCHAR(50),
PK8 VARCHAR(50),
PK9 VARCHAR(50),
IS_STRUCTURE_NAME VARCHAR(50) DEFAULT '' NOT
NULL,
VERB VARCHAR(10) DEFAULT '' NOT NULL,
PRIORITY NUMERIC(10, 0) DEFAULT 0 NOT NULL,
TIMESTAMP TIMESTAMP NOT NULL,
STATUS NUMERIC(10, 0) DEFAULT 0 NOT NULL,
RETRIEVE_TIME TIMESTAMP NOT NULL
)
```

2. **Trigger erstellen**
   Insert-Trigger überwachen, ob in der Tabelle `FATHER` Geschäftsobjekte erstellt werden.
   - **Oracle**

```
CREATE OR REPLACE TRIGGER FATHER_after_insert
AFTER INSERT ON FATHER FOR EACH ROW
BEGIN
INSERT INTO is_events (id, is_plugin_id, pk1,
pk2, pk3, pk4, pk5, pk6, pk7, pk8, pk9,
verb,is_structure_name, priority, retrieve_
time, status, timestamp)
-- '0815' ist die Plugin-ID
VALUES (SEQ_IS_EVENTS.NEXTVAL,'0815',
```

```
:new.FATHERID, NULL, NULL, NULL, NULL, NULL,
NULL, NULL, NULL,'insert', 'FATHER_CHILD', 1,
sysdate, 0, sysdate);
END;
```

- **MySQL über MySQL-Administrator**

Die Datenbankprozedur `GetNewID` muss vorhanden sein! Siehe *Datenbankprozedur getNewID für MySQL (Workbench/ Process Engine: Systemkonnektor-Guide, Kap. 7.5.2, S. 101).*

```
DELIMITER $$
DROP   TRIGGER IF EXISTS FATHER_after_insert $$
CREATE TRIGGER FATHER_after_insert  AFTER
INSERT ON FATHER FOR EACH ROW
BEGIN
DECLARE newEventID INT;
CALL getNewID('IS_EVENTS','ID',newEventID );
INSERT INTO is_events (id,is_plugin_id, pk1,
pk2, pk3, pk4, pk5, pk6, pk7, pk8, pk9,
verb,is_structure_name, priority, retrieve_
time, status,timestamp)
-- '0815' ist die Plugin-ID!
VALUES (newEventID, '0815', NEW.FATHERID, NULL,
NULL, NULL, NULL, NULL, NULL, NULL, NULL,
'insert', 'FATHER_CHILD', 1, now(), 0, now());
END $$
DELIMITER ;
```

3. **DBO Input Connector erstellen und konfigurieren**:
   a. Aktivieren Sie den Scheduler, um das Abfrage-Intervall festzulegen.
   b. Konfigurieren Sie im *Dialog „Datenbankeinstellungen“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 7.6.1, S. 102)* die Datenbankverbindung und testen Sie diese.
   c. Geben Sie die Plugin-ID des DBO Medium Connectors an, durch den die Events ausgelöst wurden.
   d. Markieren Sie die Option „Event-Listener“, um die Funktion des Konnektors festzulegen.
   e. Geben Sie den Namen der Eventtabelle an.
   f. Geben Sie eine Archiv-Tabelle an, wenn verarbeitete Events archiviert werden sollen.
4. Klicken Sie auf „Weiter“. Ein weiterer Dialog wird angezeigt.
5. Klicken Sie auf „Fertig stellen“, um die Konfiguration des Konnektors zu beenden.

7.3 Event-Archiv mit Archiv-Prozessor überwachen

Als Archiv-Prozessor schützt ein DBO Input Connector ein Event-Archiv vor dem Überlauf.

Der Konnektor prüft im angegebenen Intervall, ob die Event-Tabelle Events enthält, die älter sind als zulässig. Wenn solche Events vorhanden sind, löscht der Konnektor diese aus dem Archiv und gibt sie als Ausgangsnachricht aus.

So gehen Sie vor

1. Erstellen Sie einen DBO Input Connector.
2. Aktivieren Sie den Scheduler, um das Abfrage-Intervall für die Event-Archiv-Tabelle festzulegen.
3. Konfigurieren Sie im *Dialog „Datenbankeinstellungen“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 7.6.1, S. 102)* die Datenbankverbindung und testen Sie diese.
4. Geben Sie die Plugin-ID des DBO Input Connectors an, der die Events überwacht und archiviert.
5. Markieren Sie die Option „Archiv-Prozessor“, um die Funktion des DBO Connectors festzulegen.
6. Geben Sie den Namen der Event-Archiv-Tabelle an, die überwacht werden soll.
7. Geben Sie im Feld „Archiv-Fenster“ an, wie lange Events max. in der Event-Archiv-Tabelle aufbewahrt werden sollen.
8. Klicken Sie auf „Weiter“. Ein weiterer Dialog wird angezeigt.
9. Klicken Sie auf „Fertig stellen“, um die Konfiguration des Konnektors zu beenden.

7.4 Konfigurationsdatei: Feldbeschreibungen

Element	Attribute Name		Bedeutung
`DBStructure Description`	`DBStructureName`	Obligatorisch	Eindeutiger Name des Elements.
`DBStructure Description`	`DeleteObsoleteChilds`	Obligatorisch	Obsolete Kindelemente sollen gelöscht werden
`DBStructure Description`	`SoftDelete`	Obligatorisch	Noch nicht unterstützt.

Element	Attribute Name		Bedeutung
`DBStructure Description`	`AutoInserUpdateConversion`	Obligatorisch	Noch nicht unterstützt.
`Table`	`DBTableName`	Obligatorisch	Tabellenname in der DB
`Table`	`TagTableName`	Obligatorisch	Tabellenname in XML
`Table`	`DBProc_InsteadUpdate`	Implizit	StoredProcedure statt Update
`Table`	`DBProc_InsteadInsert`	Implizit	StoredProcedure statt Insert
`Table`	`DBProc_InsteadSelect`	Implizit	StoredProcedure statt Select
`Table`	`DBProc_InsteadDelete`	Implizit	StoredProcedure statt Delete
`Table`	`DBSoftDeletField`	Implizit	Noch nicht unterstützt.
`Table`	`DBSOftDeletValue`	Implizit	Noch nicht unterstützt.
`Field`	`DBFieldName`	Obligatorisch	Feldname in der DB
`Field`	`TagFieldName`	Obligatorisch	Feldname im XML
`Field`	`DBSPParamIdx`	Implizit	StoredProcedure Parameter Index
`Field`	`DBFieldType`	Implizit	Feldtyp
`Field`	`DBFieldLength`	Implizit	Feldlänge
`Field`	`DBIsPrimaryKey`	Obligatorisch	Primärschlüssel
`Field`	`IsAutoField`	Implizit	Autoident-Feld; wenn true, dann wird über `getNewID` eine ID geholt → Siehe *Datenbankprozedur getNewID erstellen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 7.5, S. 99).*
`Field`	`DBForeignKey_FieldName`	Implizit	Referenz auf Fremdschlüssel
`Field`	`isRequired`	Obligatorisch	Wenn `true`, dann muss die Spalte im XML vorhanden sein.
`Field`	`DefaultValue`	Implizit	Standardwert des Feldes
`Field`	`SelectExpression`	Implizit	„=“, „>“, „<„, „>=“, „<=“, „like “ (Leerzeichen danach!)

7.5 Datenbankprozedur getNewID erstellen

Dieser Abschnitt erläutert die folgenden Themen:

- *Datenbankprozedur getNewID für Oracle, S. 100*
- *Datenbankprozedur getNewID für MySQL, S. 101*

Beim Erzeugen eines neuen Geschäftsobjekts (INSERT) ruft der DBO Connector für alle Tabellenspalten, welche in der Konfigurationsdatei das Attribut `IsAutoField=true` haben, die Datenbankprozedur `getNewID()` auf, um eine eindeutige ID für die Tabellenspalten vergeben zu können.

`getNewID()` muss folgender Struktur entsprechen:

```
getNewID (i_tablename in varchar2, i_fieldname in
varchar2, o_newid out number);
```

`getNewID()` steuert anhand der übergebenen Input-Parameter (aktueller Tabellenname und Name der Tabellenspalte) die richtige Datenbank-Sequenz an und übergibt die neue ID an den DBO Connector, damit dieser das INSERT mit einer neuen ID durchführen kann.

Die folgenden Implementierungsvorschläge für `getNewID()` können sich bei einem Release-Wechsel ändern. Sie können jedoch auch nach einem Release-Wechsel die bisherige Implementierung der Prozedur beibehalten oder alternativ eine selbst entwickelte Prozedur nutzen.

7.5.1 Datenbankprozedur getNewID für Oracle

Listing

```
 1 create or replace procedure getNewID (
 2  i_tablename in  varchar2,
 3  i_fieldname in  varchar2,
 4  o_newid     out number
 5 ) is
 6   i_dotposition integer;
 7   s_owner varchar2(64);
 8   s_table varchar2(64);
 9 begin
10 i_dotposition := instr(i_tablename,'.');
11  if i_dotposition > 0 then
12     s_owner := substr(i_tablename, 1, i_dotposition - 1);
13     s_table := substr(i_tablename, i_dotposition + 1);
14  else
15     s_owner := null;
16  end if;
17 begin
18     if s_owner is null then
19       execute immediate 'select seq_' || i_fieldname || '.nextval from
   dual' into o_newid;
20      else
21       execute immediate 'select ' || s_owner || '.seq_' || i_fieldname ||
   '.nextval from dual' into o_newid;
22      end if;
23     return;
24  exception
25   when others then
26  o_newid := null;
27  end;
28 begin
29     if s_owner is null then
30       execute immediate 'select seq_' || i_tablename || '.nextval from
   dual' into o_newid;
31      else
32       execute immediate 'select ' || s_owner || '.seq_' || s_table ||
   '.nextval from dual' into o_newid;
33      end if;
34     return;
35  exception
36   when others then
```

/Listing (Abschnitt 1 von 2)

Listing

```
37 o_newid := null;
38  end;

39  if s_owner is null then
40      raise_application_error(-20000,'getNewID: Sequence seq_' || i_
   fieldname || ' or seq_' || i_tablename || ' not found');
     else
41      raise_application_error(-20000,'getNewID: Sequence ' || s_owner ||
   '.seq_' || i_fieldname || ' or ' || s_owner || '.seq_' || s_table || '
   not found');
42  end if;
43
44 end;
```

/Listing (Abschnitt 2 von 2)

7.5.2 Datenbankprozedur getNewID für MySQL

Da MySQL keine Sequenzen kennt, wird bei MySQL eine interne Sequence-Tabelle `INUBIT_DBO_SEQUENCES` verwendet:

Listing

```
 1 DELIMITER $$
 2 DROP PROCEDURE IF EXISTS getNewID $$
 3 CREATE PROCEDURE getNewID(
 4 in  i_tablename   text,
 5 in  i_fieldname   text,
 6 out o_newid       int
 7 )
 8 BEGIN
 9 DECLARE entryCount INT;
10 SELECT COUNT(*) INTO entryCount FROM INUBIT_DBO_SEQUENCES WHERE
   SEQUENCE_TABLE=i_tablename AND SEQUENCE_COLUMN = i_fieldname;
11 IF entryCount=0 THEN /* create entry  */
12 INSERT INTO INUBIT_DBO_SEQUENCES (SEQUENCE_TABLE,SEQUENCE_
   COLUMN,SEQUENCE_VALUE) Values (i_tablename,i_fieldname,0);
13 SET o_newid  = 0;
14 ELSE  /* increase sequence */
```

/Listing (Abschnitt 1 von 2)

Listing

```
15 UPDATE INUBIT_DBO_SEQUENCES SET SEQUENCE_VALUE=last_insert_id(SEQUENCE_
   VALUE+1) WHERE SEQUENCE_TABLE=i_tablename AND SEQUENCE_COLUMN = i_
   fieldname;
16 SET o_newid  = last_insert_id();
17 END IF;
18 END $$
19 DELIMITER;
20 DROP TABLE IF EXISTS INUBIT_DBO_SEQUENCES;
21 CREATE TABLE INUBIT_DBO_SEQUENCES(
22 SEQUENCE_TABLE   VARCHAR(64)          NOT NULL,
23 SEQUENCE_COLUMN  VARCHAR(64)          NOT NULL,
24 SEQUENCE_VALUE       INT(10) UNSIGNED NOT NULL,
25 PRIMARY KEY(SEQUENCE_TABLE ,SEQUENCE_COLUMN )
26 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

/Listing (Abschnitt 2 von 2)

7.6 Dialogbeschreibungen

Dieser Abschnitt erläutert die folgenden Themen:

- *Dialog „Datenbankeinstellungen“, S. 102*
- *Dialog „Datenbank Connection Pooling“, S. 105*
- *Dialog „DBO-Konfigurationsdatei“, S. 105*

7.6.1 Dialog „Datenbankeinstellungen“

Datenbankverbindung

- **Voreinbustellung**
  Zur Auswahl einer vorkonfigurierten Datenbank, um die Felder „Datenbank-URL“ und „Datenbanktreiber-Klasse“ sinnvoll zu belegen.
  Bei Auswahl von „IS Log Database“ verbindet sich der Database Connector mit der Logging/Monitoring-Datenbank der inubit Suite 6. Für die Verbindung wird die Konfiguration der inubit Process Engine verwendet, deswegen werden alle anderen Felder in diesem Bereich deaktiviert.

Falls Ihre Datenbank nicht vorhanden ist, wählen Sie die Voreinstellung „Custom“ und füllen Sie die Felder manuell aus.

Unter Windows können Sie auf die JDBC-ODBC-Bridge ausweichen.

Wählen Sie die JDBC-ODBC-Bridge nur, wenn Sie keinen JDBC-Zugriff auf Ihrer Datenbank einrichten können. Eine JDBC-ODBC-Bridge hat eine schlechtere Performance, der Funktionsumfang kann eingeschränkt sein und die automatische Typerkennung ist deutlich langsamer.

- **Datenbank-URL**
  Wird vorbelegt, wenn Sie eine vorkonfigurierte Datenbank gewählt haben.
  Geben Sie den Host, den Port und den Datenbanknamen an.
- **Datenbanktreiber-Klasse**
  Entspricht der JDBC-Verbindung der eingesetzten Datenbank. Um eine andere Datenbank zu verwenden, müssen Sie einen passenden Treiber installieren.
  → Siehe *Treiber installieren (Process Engine: Administrator- und Entwickler-Guide, Kap. 3.4, S. 51).*

Nach Installation eines JDBC-Treiber für den MS SQL Server müssen Sie die inubit Process Engine neu starten.

Um eine Adabas Datenbank einzusetzen, kopieren Sie die Treiberklasse direkt in das Verzeichnis `<iS-installdir>/server/JBoss/server/default/lib` bzw. `<iS-installdir>/server/webapps/ibis/WEB-INF/lib`.

- **Benutzer/Passwort**
  Eingabe ist abhängig von Ihrer Datenbank.
- **Connection Pooling/Button „Einstellungen“**
  Aktiviert das Connection Pooling. Mit dem Button öffnen Sie den *Dialog „Datenbank Connection Pooling“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 6.5.5, S. 87)* zur Konfiguration des Connection Pooling.
- **Spezielle Verbindungsparameter**
  Zum Erstellen von Parametern wie z. B. Timeouts, Anzahl offener Verbindungen etc., die genutzt werden, um die Datenbankverbindung abhängig von Ihrer verwendeten Datenbank zu konfigurieren. Die Parameter definieren Sie als Name-Wert-Paare.

Spezielle Einstellungen

- **Plugin-ID**
  - Bei einem DBO Medium Connector:

Die ID identifiziert den jeweiligen Konnektor und muss innerhalb der inubit Suite 6 und pro Datenbank eindeutig sein.

- Bei einem DBO Input Connector:

  ID des DBO Medium Connectors, der die Events erzeugt, welche der DBO Input Connector überwachen soll. Ein DBO Input Connector kann nur die Events von einem DBO Medium Connector empfangen, der dieselbe ID wie der DBO Input Connector selbst hat.

Beispiel: Sie nutzen ein Szenario, in dem ein DBO Medium Connector Geschäftsobjekte erstellt oder modifiziert, ein DBO Input Listener diese Events überwacht und ein weiterer DBO Input Listener die Event-Archiv-Tabelle überwacht. In diesem Szenario müssen alle drei Konnektoren dieselbe ID haben.

- **Event-Listener**/**Archiv-Prozessor**

  (Nur beim Input Connector)

  Legt die Funktion des Input Connectors fest.

  → Siehe

  - *Events mit Event-Listener überwachen und archivieren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 7.2, S. 94)*
  - *Event-Archiv mit Archiv-Prozessor überwachen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 7.3, S. 97)*

- **Max. Anzahl von Events pro Aufruf**

  (Nur bei einem Input Connector)

  Mit der Angabe legen Sie fest, wie oft der Workflow pro Aufruf maximal gestartet werden soll. Mit der Begrenzung können Sie die Systemauslastung der inubit Process Engine steuern.

  Der Workflow wird so oft gestartet, bis entweder

  - keine weiteren Events zu holen sind
  - oder die maximale Anzahl von Events erreicht ist.

Im Test-Modus wird der Workflow bei einem Aufruf genau einmal gestartet und liest exakt ein Event.

- **Event-Tabelle**

  (Nur beim Input Connector)

  Name der Tabelle, aus welcher Events geholt werden sollen.

- **Event-Archiv-Tabelle**

  (Nur beim Input Connector)

  Name der Tabelle, in der verarbeitete Events archiviert werden. Wenn keine Tabelle angegeben ist, dann werden Events nicht archiviert.

- **Archiv-Fenster in Tagen**

  (Nur beim Input Connector)

  Alle archivierten Events, die älter als X Tage sind, werden aus dem Archiv gelöscht und dem Workflow zur Verfügung gestellt.

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

7.6.2 Dialog „Datenbank Connection Pooling“

→ Siehe *Dialog „Datenbank Connection Pooling“ (Workbench/ Process Engine: Systemkonnektor-Guide, Kap. 6.5.5, S. 87).*

7.6.3 Dialog „DBO-Konfigurationsdatei“

In diesem Dialog wird die Konfigurationsdatei des DBO Connectors angezeigt. In der Konfigurationsdatei wird die Struktur der Datenbanktabellen auf eine XML-Struktur abgebildet.

→ Siehe *Konfigurationsdatei: Feldbeschreibungen (Workbench/ Process Engine: Systemkonnektor-Guide, Kap. 7.4, S. 97).*

Beim Erstellen eines DBO Connectors wird automatisch ein zur gewählten Datenbank passendes Template für die Konfigurationsdatei geladen. Dieses Template müssen Sie anpassen.

Sie können Ihre angepasste Konfigurationsdatei im Repository oder im Dateisystem speichern, um diese in weiteren DBO Connectoren wiederzuverwenden.

8 Exchange Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Der Exchange Connector verbindet einen Exchange Server über Intranet/Internet mit der inubit Suite 6.

Mit Hilfe des Exchange Connectors können Sie folgende Funktionen des Exchange Servers nutzen:

- **Mails abholen/versenden**
  Es können alle, eine bestimmte Anzahl Mails oder nur ungelesene Mails abgeholt werden.
- **Aufgaben abholen/erstellen**
- **Postfächer auslesen**
  Erzeugt eine Liste aller existierenden Postfächer und liest Abwesenheitsnotizen aus (wenn vorhanden).
- **Kontakte auslesen/anlegen**
- **Verfügbarkeit abfragen**
- **Kalendereinträge abholen**
- **Termine anlegen**
- **Einträge löschen**
  Einträge können sofort gelöscht oder in den Ordner „gelöschte Objekte“ verschoben werden.
- **Einträge verschieben**
  Einträge können in andere Ordner verschoben werden, z. B. E-Mails von der Inbox in einen anderen Ordner.

→ Siehe auch

  - *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
  - *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
  - *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

8.1 Installationsvoraussetzungen

Sie müssen die Microsoft Exchange Server MAPI Client and Collaboration Data Objects (CDO) und J-Integra® for Exchange installieren, um die Kommunikation mit dem Exchange Server herzustellen.

Diese Software erhalten Sie zusammen mit der Lizenz für den Exchange Connector von der inubit AG.

Detaillierte Installationsanleitungen finden Sie unter http://j-integra.intrinsyc.com/support/exchange/doc/.

Um einen stabilen Betrieb sicherzustellen, empfiehlt inubit AG, das CDO auf einem anderen Server als dem Exchange Server zu installieren:

Beachten Sie außerdem:

- Der CDO-Server und der Exchange-Server müssen in derselben Domain sein.
- Auf dem CDO-Server darf Outlook **nicht** installiert sein! Outlook besitzt eine eigene CDO-Installation, die Konflikte verursachen kann.
- Mehrere Exchange-Konnektoren dürfen nicht parallel auf dieselbe CDO-Installation zugreifen! Das CDO ist nicht multithreading-sicher.

8.2 Exchange Connector verwenden

Die von Ihnen konfigurierte Aktion bestimmt, wie der Exchange Connector in einem Technical Workflow verwendet wird. Für die meisten Aktionen benötigt der Exchange Connector eine XML-basierte Eingangsnachricht, in der die auszuführende Aktion und deren Parameter definiert sind.

Die XML-basierte Eingangsnachricht erstellen Sie mit einem der mitgelieferten Templates in einem XSLT Converter, den Sie mit den Exchange Connector verbinden.

Die folgende Tabelle gibt für jede Aktion an, ob ein Template nötig ist, und wenn ja, welches:

Aktion	Template
Mails abholen	Keine Eingangsnachricht
Mails versenden	E-Mails müssen in einem gültigen MIME-Format vorliegen. Repository > `Global > System > Mapping Templates > MIME Adapter` (Template für IBISMIME-Format)
Postfächer auslesen	Keine Eingangsnachricht
Aufgaben abholen/erstellen	Repository > `Global > System > Mapping Templates > Exchange Connector, GetTasks.xsd` bzw. `CreateTasks.xsd`
Kontakte auslesen/anlegen	Repository > `Global > System > Mapping Templates > Exchange Connector, GetContacts.xsd` bzw. `CreateContacts.xsd`
Verfügbarkeit abfragen	Repository unter `Global > System > Mapping Templates > Exchange Connector > GetFreeBusy.xsd`
Kalendereinträge abholen/ Termine anlegen	Repository > `Global > System > Mapping Templates > Exchange Connector, GetAppointments.xsd` bzw. `CreateAppointments.xds.`
Einträge löschen	Repository > `Global > System > Mapping Templates > Exchange Connector> DeleteItems.xsd`
Einträge verschieben	Repository > `Global > System > Mapping Templates > Exchange Connector> MoveItems.xsd`

Zugriff auf mehrere Postfächer

Jeder Exchange Connector kann auf genau ein Postfach zugreifen; um mehrere Postfächer auszulesen, verwenden Sie das Variablen Mapping. Damit können Sie die Eigenschaft `exchange.getmails.mailbox` des Exchange Connectors (zum Abholen von Mails) dynamisch überschreiben.

→ Siehe *Workflow-Variablen und Mappings (Workbench: Benutzer-Guide, Kap. 14, S. 365).*

8.3 Dialogbeschreibungen

Dieser Abschnitt erläutert die folgenden Themen:

- *Dialog „Einträge verschieben“, S. 112*
- *Weitere Aktionen konfigurieren, S. 112*

8.3.1 Dialog „Allgemeine Einstellungen“

Dieser Dialog bietet folgende Optionen:

Einstellungen

- **Domäne**
  Name der Verwaltungsstruktur im Windows-Netzwerk, an der Sie sich mit den Zugangsdaten aus den Feldern „Benutzer“/“Passwort“ anmelden möchten.
- **CDO Server**
  IP-Adresse oder Hostname des Microsoft Windows Servers, auf dem Microsoft CDO und J-Integra® for Exchange installiert sind.
- **Exchange Server**
  IP-Adresse bzw. Hostname des Microsoft Exchange Servers, zu dem die Verbindung über das CDO aufgebaut werden soll.
- **Benutzer/Passwort**
  Zugangsdaten des Benutzers, mit denen Sie sich an der oben genannten Domäne anmelden möchten. Der Zugang muss über die Berechtigungen verfügen, welche für die aktuelle Aktion nötig sind.

Aktion

Wählen Sie eine der möglichen Aktionen aus.
→ Siehe *Verwendung (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 8, S. 107)*

8.3.2 Dialog „Mails abholen“

Dieser Dialog bietet folgende Optionen:

Nachrichten abholen

- **Mailbox**
  Postfach, auf welches sich die Aktion bezieht.
- **Lesen**
  Wählen Sie aus, ob alle oder nur die ungelesenen Mails abgeholt werden sollen.

- **Nach der Verarbeitung**
  Geben Sie an, ob E-Mails
  - vom Exchange Server abgeholt und dabei vom Exchange Server gelöscht werden sollen
  - oder vom Exchange Server abgeholt und die Original-E-Mails als „Gelesen" markiert werden sollen.
  - nicht weiter bearbeitet werden sollen.

Anzahl von Nachrichten pro Aufruf

Wählen Sie eine der Optionen:

- **Alle**
  Während einer Modul-Ausführung werden alle Mails in der angegebenen Mailbox vom Exchange Server geholt.
- **Maximum**
  Während einer Modul-Ausführung wird nur die angegebene Anzahl von Mails abgeholt.

Filtereinstellungen

Zum Definieren von Filterkriterien, denen E-Mails entsprechen müssen, damit diese vom Mail-Server abgeholt werden:

- **Verknüpfung der Filter**
  Wenn mehr als ein Filter angegeben ist:
  - UND
    Alle Filter müssen auf eine Nachricht zutreffen, damit diese geholt wird.
  - ODER
    Mindestens ein Filter muss auf eine Nachricht zutreffen, damit diese geholt wird.
- **Filter hinzufügen**
  Der Button fügt folgende Zeile ein:

  In dieser Zeile definieren Sie die Filterkriterien:
  - Subject/From/To: Liste zur Auswahl des Feldes, das gefiltert werden soll.
  - enthält: Angabe darüber, dass im gewählten Feld eine bestimmte Zeichenkette vorhanden sein muss.
  - Eingabefeld: Zur Eingabe der Zeichenkette, die vorhanden sein soll.

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

8.3.3 Dialog „Einträge verschieben“

Dieser Dialog bietet folgende Optionen:

Einträge verschieben

- **Mailbox**
  Postfach bzw. Zugangskennung, auf welches sich die Aktion bezieht.
- **In Ordner**
  Name des Ordners bzw. Pfad zum Ordner, in welchen der Eintrag verschoben werden soll. Pfadangaben müssen nach folgendem Muster angegeben sein: [`Ordnername] >>> [Ordnername]`.

Verbindungstext

Verbindung testen

Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

8.3.4 Weitere Aktionen konfigurieren

Für alle Aktionen (außer „Mails abholen“) wird ein Dialog angezeigt.

Im Dialog zum Auslesen der Postfächer muss nur das Postfach angegeben werden, auf welches sich die Aktion bezieht.

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

9 Execution Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

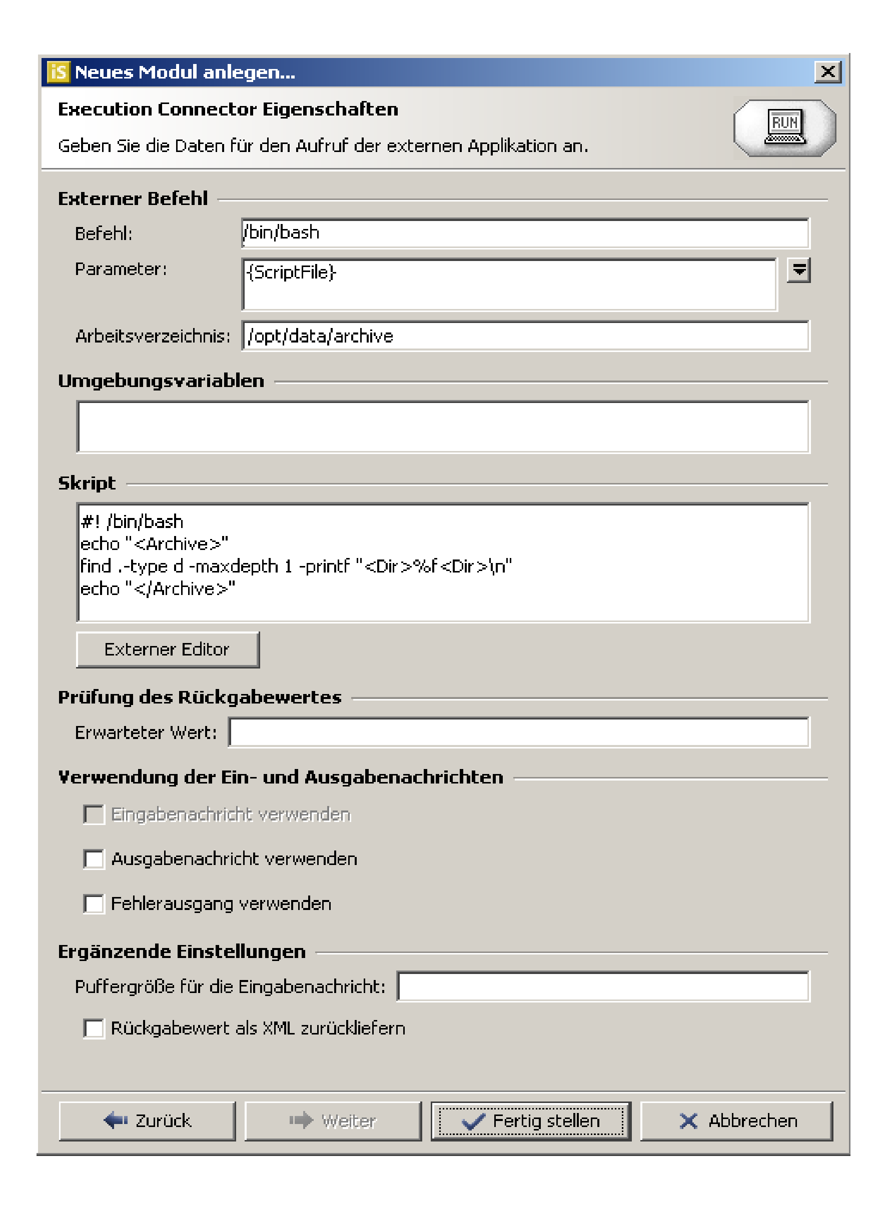
Mit dem Execution Connector rufen Sie eine externe Anwendung innerhalb eines Workflows auf. Sie können jede Anwendung aufrufen, welche über die Kommandozeile gestartet werden kann.

→ Siehe auch

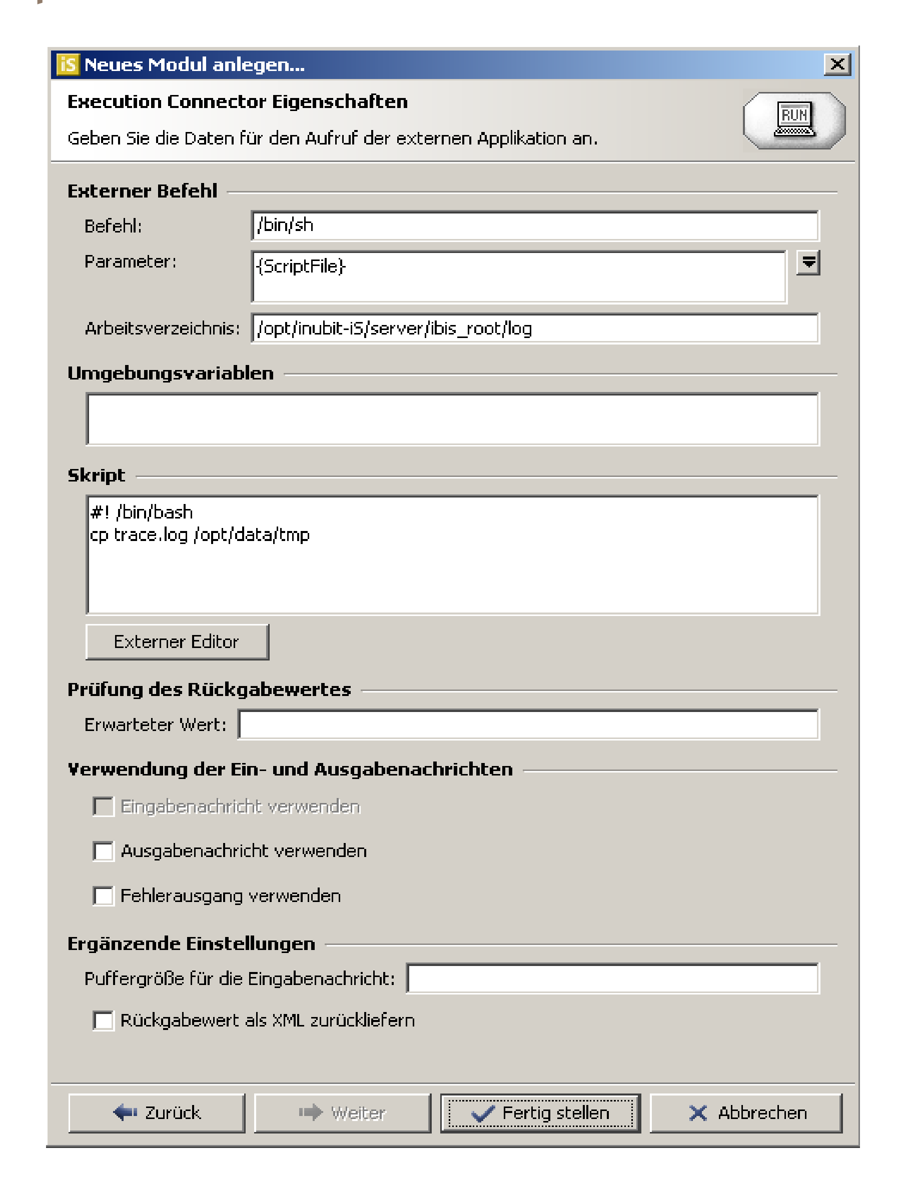
- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

9.1 Beispiel: Verzeichnis ausgeben

Mit der folgenden Konfiguration wird der Inhalt des Verzeichnisses `/opt/data/archive` gelistet und über die echo-Anweisungen in eine XML-Struktur gepackt:

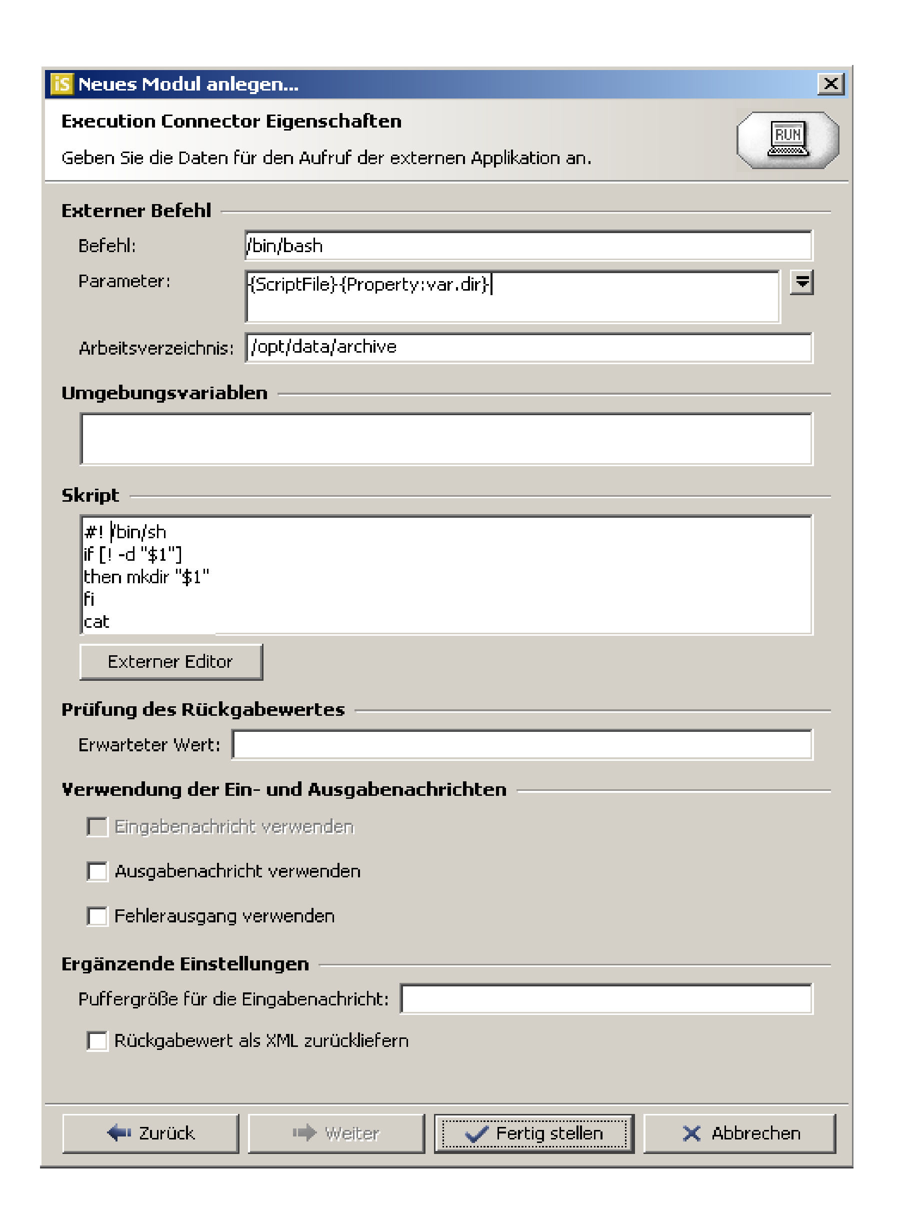

Neues Modul anlegen...
Execution Connector Eigenschaften
Geben Sie die Daten für den Aufruf der externen Applikation an.
RUN
Externer Befehl
Befehl:
/bin/bash
Parameter:
{ScriptFile}
Arbeitsverzeichnis:
/opt/data/archive
Umgebungsvariablen
Skript
#! /bin/bash
echo "<Archive>"
find .-type d -maxdepth 1 -printf "<Dir>%f<Dir>\n"
echo "</Archive>"
Externer Editor
Prüfung des Rückgabewertes
Erwarteter Wert:
Verwendung der Ein- und Ausgabenachrichten
Eingabenachricht verwenden
Ausgabenachricht verwenden
Fehlerausgang verwenden
Ergänzende Einstellungen
Puffergröße für die Eingabenachricht:
Rückgabewert als XML zurückliefern
Zurück
Weiter
Fertig stellen
Abbrechen

9.2 Beispiel: Datei kopieren



9.3 Beispiel: Verzeichnis anlegen

In diesem Beispiel wird auf die Workflow-Variable `var.dir` zugegriffen. Wenn kein Verzeichnis mit dem Namen in `var.dir` existiert, wird es angelegt:



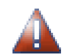
9.4 Dialog „Execution Connector Eigenschaften“

In diesem Dialog konfigurieren Sie den Aufruf der externen Applikation.

Externer Befehl

- **Befehl**
  Pfad und Name der aufzurufenden Applikation.

Die aufzurufende Applikation muss auf demselben Rechner installiert sein wie der Applikationsserver (JBoss/Tomcat). Der Pfad kann relativ zum Applikationsserver oder absolut angegeben werden.



Nicht-termininierende Befehle gefährden die Stabilität der inubit Process Engine! Beispiel: `cmd` ohne `/C` zum Beenden des Befehles nach der Ausführung.

- **Parameter**

  Parameter, mit denen die externe Applikation aufgerufen werden soll. Mehrere Argumente trennen Sie durch Leerzeichen.

  Über den Button rechts neben dem Argumente-Feld können Sie folgende Platzhalter eingeben:

  - **`{File}`**

    Um die Eingangsnachricht als temporäre Datei an die aufgerufene Applikation zu übergeben. Nach ihrer Ausführung schreibt diese Applikation das Ergebnis in dieselbe Datei.

Nutzen Sie `{inputFile}` und `{outputFile}`, wenn zwei verschiedene Dateien verwenden werden sollen.
`{File}` und `{InputFile}` schließen sich gegenseitig aus!

  - **`{XPath:Pfadangabe}`**

    Übergibt einen Knotenwert aus der XML-Eingangsnachricht an die aufgerufene Applikation.

    Ein Klick auf diesen Parameter öffnet den XPath-Assistenten, mit dem Sie XPath-Ausdrücke zum Auslesen des Knotenwertes erstellen können.

    → Siehe *XPath-Assistent (Workbench: Benutzer-Guide, Kap. 1.16, S. 67).*

  - **`{Property:Propertyname}`**

    Übergibt eine Moduleigenschaft.

  - **`{ScriptFile}`**

    Übergibt das Skript aus dem Feld „Script“.

  - **`{InputFile}`**

    Übergibt eine Eingangsnachricht an die aufgerufene Applikation, ohne diese nach der Ausführung wieder auszulesen, z. B.

    - Befehl: `cmd.exe`
    - Argumente: `/C {ScriptFile.cmd} {InputFile}`
    - Script: `type "%1" > inputMessageDuplicated.dat`

  - **`{OutputFile}`**

Übergibt der aufgerufenen Applikation eine Datei, in welche die Applikation nach ihrer Ausführung das Ergebnis schreibt. Dieser Platzhalter wird meist zusammen mit `{ScriptFile}` genutzt, z. B.:

- Befehl: `cmd.exe`
- Argumente: `/C {ScriptFile.cmd} {OutputFile}`
- Script: `echo "This will be streamed into outputfile and provided to workflow" > "%1"`

Bei den Argumenten `{File}`, `{ScriptFile}`, `{InputFile}` und `{OutputFile}` können Sie eine der Dateiendung `.cmd` bzw. `.bat` angeben, z. B. `{ScriptFile.bat}`.

- **Arbeitsverzeichnis**

  Für temporäre Dateien. Der Execution Connector und einige Applikationen benötigen ein solches Verzeichnis. Wenn Sie kein Verzeichnis angeben, wird das jeweilige Standard-Verzeichnis unter Tomcat bzw. JBoss verwendet.

  Das Standard-Arbeitsverzeichnis ist `Tomcat\bin` bzw. `JBoss\bin` oder das des jeweils genutzten Applikationservers, wenn dieser nicht JBoss oder Tomcat ist.

Umgebungsvariablen

Umgebungsvariablen werden als Variablen/Wertepaare (getrennt durch ein „=“ Zeichen) übergeben. Mehrere Umgebungsvariablen trennen Sie durch jeweils einen Zeilenumbruch.

Skript

- Eingabefeld

  Um ein Skript eingeben zu können, müssen Sie im Feld „Argumente“ den Parameter `{ScriptFile}` einfügen.

  Um auf die Parameter zuzugreifen, die im Feld „Argumente“ angegeben sind, verwenden Sie Variablen. Deren Syntax ist abhängig vom eingesetzten Betriebssystem:

  - Windows: `%1`
  - Linux: `$1`

- Button „Externer Editor“

  Zum Editieren des Skripts. Öffnet den externen Editor, der im Applikationsprofil „Generic Application“ bzw. „Script Editor“ angegeben ist.

  → Siehe *Applikationsprofile (Process Engine: Administrator- und Entwickler-Guide, Kap. 11.1, S. 135).*

Prüfung des Rückgabewertes

Erwarteter Wert

Numerischer Wert.

- Wenn der angegebene und der tatsächliche Rückgabewert gleich sind, dann wird die Workflow-Ausführung fortgesetzt.
- Wenn ein anderer Wert als der angegebene zurückgegeben wird, dann wird der Fehlercode der aufgerufenen Applikation XML-formatiert (z. B. `<ReturnValue>value</ReturnValue>`) an den Workflow übergeben.
- Wenn kein erwarteter Wert angegeben ist, läuft der Workflow grundsätzlich weiter.

Um sicherzustellen, dass fehlerhafte Ausführungen immer an den Workflow gemeldet werden, geben Sie 0 ein, denn Fehlercodes sind immer ungleich Null.

Verwendung der Ein- und Ausgangsnachrichten

- **Eingabenachricht verwenden**
  (nur beim Medium/Output Connector)
  Wenn markiert, dann wird die Eingangsnachricht des Execution Connectors an die externe Applikation weitergeleitet.
- **Ausgabenachricht verwenden**
  (nur beim Input/Medium Connector)
  Wenn markiert, dann wird die Ausgangsnachricht der externen Anwendung an den Execution Connector übergeben und von diesem an den Workflow weitergeleitet.
- **Fehlerausgang verwenden**
  (nur beim Input/Medium Connector)
  Wenn markiert, dann wird die Fehler-Ausgabe der externen Anwendung an den Execution Connector und danach in den Workflow übergeben.

Ergänzende Einstellungen

- **Puffergröße für die Eingabenachricht**
  Anzahl der Bytes für die Speicherreservierung. Wenn Sie keine Angabe machen, wird der Standard 4096 verwendet.
  Ein großer Wert verbraucht viel Speicher, der zum Aufrufzeitpunkt angefordert wird, und beschleunigt die Übergabe der Eingabenachricht an die aufrufende Applikation.
  Ein kleiner Wert verbraucht weniger Speicher. Die Übergabe der Eingangsnachrichten kann jedoch Zeit kosten, wenn diese viel größer als der Puffer sind und daher in Paketen mit der Größe des Puffers übermittelt werden müssen.
- **Rückgabewert als XML zurückliefern**

Nur sinnvoll in Kombination mit der Option „Ausgabenachricht verwenden“!

Der Rückgabewert wird in eine XML-Struktur eingebettet und zurückgeliefert.

10 File Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Ein File Connector sorgt für die Kommunikation der inubit Process Engine mit dem Dateisystem.

Konnektortypen

Sie haben folgende Konfigurationsmöglichkeiten:

- **File Input Connector**
  Holt Dateien und Verzeichnisse aus dem Dateisystem der inubit Process Engine und übergibt sie an das nächste Modul zur Bearbeitung.
- **File Output Connector**
  Schreibt das Ergebnis des vorhergehenden Moduls in eine Datei in einem Verzeichnis der inubit Process Engine. Wenn das Ergebnis ein Zip-Archiv ist, dann entpackt der File Output Connector dieses vor dem Speichern.

→ Siehe auch
  - *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
  - *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
  - *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

10.1 Modulvariablen des File Connectors

Bei der Ausführung eines File Connectors werden folgende Variablen gesetzt, die Sie an den nachfolgenden Modulen des Workflow auswerten oder überschreiben können:

- **`ReadFileDate`**
  Letztes Änderungsdatum der Eingangsdatei, Format `dd.mm.yyyy hh:mm:ss`.
- **`ReadFileName`**
  Name der Eingangsdatei mit Extension.
- **`ReadFileSize`**
  Größe der Eingangsdatei, in Bytes.
- **`ReadFileWildcardname`**

Gibt den Inhalt der Wildcard aus, die beim Konfigurieren des File Connectors im Dateinamen verwendet wurde.

- **`WriteFileDate`**
  Nur verfügbar beim Output Connector, wenn als Eingangsformat „Daten“ gewählt wurde!
  Erzeugungsdatum der Ausgabedatei, Format `dd.MM.yyyy HH:mm:ss`.
- **`WriteFileName`**
  Nur verfügbar beim Output Connector, wenn als Eingangsformat „Daten“ gewählt wurde!
  Name der Ausgabedatei.
- **`WriteFileWildcardname`**
  Nur verfügbar beim Output Connector, wenn als Eingangsformat „Daten“ gewählt wurde und wenn in dem Dateinamen, der erzeugt werden soll, z. B. Timestamp oder Wildcard verwendet werden!
  Gibt den Inhalt der Wildcard aus, die beim Konfigurieren des File Output Connectors im Dateinamen verwendet wurde.

→ Für Infos über Variablen und deren Verwendung siehe *Workflow-Variablen und Mappings (Workbench: Benutzer-Guide, Kap. 14, S. 365)*.

10.2 Dialogbeschreibungen

Dieser Abschnitt erläutert die folgenden Themen:

- *Dialog „Zu lesende Datei(en)“, S. 122*
- *Dialog „Datenweitergabe“, S. 125*
- *Dialog „Zu schreibende Datei“, S. 127*

10.2.1 Dialog „Zu lesende Datei(en)“

(Input Connector)

Dieser Dialog bietet folgende Optionen:

Modus

Legen Sie fest, in welchem Modus der File Connector arbeiten soll:

- **Dateimodus**

Pro Ausführung wird eine Datei im angegebenen Verzeichnis gelesen (wenn deren Name dem Muster entspricht, das im Feld „Datei > Name“ angegeben ist). Welche Datei gelesen wird, hängt von Ihren Angaben bei „Lesereihenfolge“ ab.

- **Verzeichnismodus**
  Es werden alle Dateien im angegebenen Verzeichnis gelesen (wenn deren Dateinamen dem Muster entsprechen, das im Feld „Datei > Name“ angegeben ist). Im Verzeichnismodus wird immer eine Ausgangsnachricht (XML) erzeugt, d. h. der Technical Workflow startet in jedem Fall.

Damit ist es z. B. möglich, hinter einem File Input Connector mit einem Demultiplexer o. ä. auf den jeweiligen Zustand der Datei zu reagieren.

Verzeichnis

Name

Pfad zum Verzeichnis, aus dem Dateien gelesen werden sollen. Der Pfad kann absolut oder relativ zum Verzeichnis `<iS-installdir>/server/Tomcat/bin` angegeben werden.

Datei

- **Name**
  Dateiname. Optional mit Wildcard bzw. regulärem Ausdruck.

Siehe http://www.regular-expressions.info/tutorial.html.

- **Invertieren**
  Wenn markiert, dann wird das Muster, das für den Dateinamen angegeben wurde, umgekehrt. Damit werden alle Dateien gelesen, deren Name dem Muster nicht entspricht.
- **Mit Wildcard**
  Wenn markiert, dann können Sie beim Eingeben des Dateinamens eine Wildcard (Asterisk *) verwenden oder den Dateinamen vollständig eingeben.
- **Mit regulärem Ausdruck**
  Wenn markiert, dann können Sie mit Hilfe eines regulären Ausdrucks ein Muster angeben, dem die Dateinamen der einzulesenden Dateien entsprechen müssen.

Reguläre Ausdrücke müssen in Schrägstriche eingeschlossen werden, z. B. `/(\d\d).(\d\d).(\d\d\d\d)/`.

Für Informationen über reguläre Ausdrücke siehe z. B. http://www.perl.com/doc/manual/html/pod/perlre.html oder http://www.regular-expressions.info/tutorial.html.

- **Wildcard-Inhalt des vorher ausgeführten File oder FTP Connectors verwenden**

  Beispiel: Ein vorher ausgeführter File Connector wurde so konfiguriert, dass im Feld „Dateiname“ eine Datei `Test*.xml` angegeben wurde. Als Ausgangsnachricht wurde vom File Connector eine Datei mit Namen `Test5.xml` weitergeleitet. Der Wildcard-Inhalt ist in diesem Fall das Zeichen „5“. Wenn Sie im vorliegenden Connector ebenfalls eine Wildcard angegeben haben, zum Beispiel „Ausgabe*.xml“, so wird an Stelle der Wildcard das Zeichen 5 eingefügt. Die Ausgabe des hier konfigurierten Connectors ist daher „Ausgabe5.xml“.

- **Dateien nach dem Lesen löschen/Fehler erzeugen, wenn Löschvorgang fehlschlägt**

  Wenn aktiviert, werden die gelesenen Dateien nach dem Lesen gelöscht.

  Wenn zusätzlich die Option „Fehler erzeugen, wenn Löschvorgang fehlschlägt“ aktiviert ist, dann wird im Queue Manager ein Fehlereintrag für den Workflow mit diesem Modul angezeigt, falls die Datei nicht gelöscht werden kann. Wenn ein Fehlerausgang konfiguriert ist, wird dieser durchlaufen. Sonst wird die Ausführung des Moduls abgebrochen.

- **Freigabe der Datei abwarten/Prüfintervall in Sekunden**

  (Nur im Dateimodus)

  Markieren Sie diese Option, um sicherzustellen, dass der File Input Connector keine Dateien liest, während diese noch von anderen Applikationen geschrieben werden. Wenn die Option markiert ist, dann prüft der File Input Connector in einem konfigurierbaren Intervall die Zeitstempel und Größen der zu lesenden Dateien und vergleicht diese. Die Dateien werden nur eingelesen, wenn sich die Angaben nicht geändert haben.

- **Max. Anzahl v. Ausführungen pro zeitgesteuertem Aufruf**

  Mit der Angabe legen Sie fest, wie oft pro auslösendem Event der Workflow maximal gestartet werden soll. Auslösendes Event ist das Erreichen eines Zeitpunktes. Dann wird der Workflow so oft gestartet, bis entweder

  - keine weiteren Daten zu holen/zu senden sind oder
  - die Anzahl erreicht wurde, welche bei „maximale Ausführungen pro Aufruf" konfiguriert wurde.

  Mit der Begrenzung der Ausführung steuern Sie die Systemauslastung der inubit Process Engine.

  Die Angabe wird im Test-Modus ignoriert, dann wird der Workflow genau einmal gestartet.

Die Option ist nur bei aktiviertem Scheduler aktiv!

Lesereihenfolge

Wenn mehr als eine Datei übertragen wird, legen Sie fest, welche Dateien zuerst übertragen werden und in welcher Reihenfolge.

10.2.2 Dialog „Datenweitergabe“

(Input Connector)

In diesem Dialog konfigurieren Sie die Ausgabe des Konnektors.

Verzeichnis Konfiguration

(Bei Auswahl „Verzeichnismodus)“

- **Inkl. Verzeichnisse**
  Wenn markiert, liest der File Connector zusätzlich zu den Dateien im angegebenen Verzeichnis auch das Verzeichnis selbst ein.
  - **mit Unterverzeichnissen**
    Wenn markiert, liest der File Connector zusätzlich rekursiv alle Unterverzeichnisse des angegebenen Verzeichnisses ein.
- **Leere Verzeichnisse nach dem Lesen löschen**
  Löscht leere Verzeichnisse nach dem Lesen; diese Option ist sinnvoll vor allem in Kombination mit der Option „Dateien nach dem Lesen löschen“.
- **Limit Dateianzahl**
  Maximale Anzahl von Dateien, die eingelesen werden.
- **Limit Gesamtgröße**
  Maximale Größe, bis zu der Dateien eingelesen werden.

Dateiprüfung

(Bei Auswahl „Dateimodus“)

- **Abbruch des Workflows, wenn die Datei nicht existiert**
  Die Ausführung des Workflows wird mit einem Fehler abgebrochen, wenn der Zugriff auf die angegebene Datei fehlschlägt.

Für den Fall, dass der File Input Connector nicht das erste Modul in einem Technical Workflow ist und die Datei nicht existiert, wird die Ausführung nicht abgebrochen, sondern mit einer leeren Nachricht fortgesetzt. Es erfolgt kein Eintrag im Log.

- **Erzeuge XML Ausgangsnachricht mit der Statusinformation darüber, dass die Datei existiert und leer ist, ansonsten Abbruch des Workflows**

Prüft, ob eine passende Datei mit einer Größe=0 vorhanden ist. Wenn ja, wird eine XML-Datei mit dem Element `<FileExists>` mit dem Wert „true“ ausgegeben. In jedem anderen Fall wird die Workflow-Ausführung mit einem Fehler abgebrochen. Die Eingabedatei wird nicht ausgegeben.

- **Erzeuge XML Ausgangsnachricht mit der Statusinformation darüber, ob die Datei existiert oder nicht**
  Prüft, ob eine geeignete Datei vorhanden ist oder nicht. Ausgegeben wird eine XML Datei mit dem Element `<FileExists>` und dem Wert „false“, wenn die angegebene Datei nicht existiert, bzw. „true“, wenn die angegebene Datei existiert. Die Eingabedatei wird nicht ausgegeben.
- **Erzeuge XML Ausgangsnachricht mit der Statusinformation darüber, dass die Datei nicht existiert, ansonsten Abbruch des Workflows**
  Prüft, ob eine passende Datei vorhanden ist. Wenn nicht, wird eine XML-Datei mit dem Element `<FileExists>` und dem Wert „false“ ausgegeben. In jedem anderen Fall wird die Workflow-Ausführung mit einem Fehler abgebrochen. Die Eingabedatei wird nicht ausgegeben.

Datenweitergabe

(Nur im Verzeichnismodus; im Dateimodus nur bei „Abbruch des Workflows, wenn Datei nicht existiert".)

- **Art**
  Legen Sie fest, wie die Eingangsnachrichten an den Workflow übergeben werden sollen:
  - **Daten** (nur bei Auswahl „Dateimodus“)
    Die Eingangsnachrichten werden unverändert weitergegeben. Diese Option ist auszuwählen, wenn die Eingangsnachrichten z. B. als XML vorliegen.
  - **IBISDirectory-Xml**
    Legt fest, dass die Eingangsnachrichten im IBISDirectory XML-Format weitergegeben werden. Dabei wird für jede Eingangsnachricht ein XML-Wrapper-Element erzeugt.
  - **Zip**
    Die Eingabedateien werden in einer Zip-Datei komprimiert.

XML Konfiguration

(nur bei „Datenweitergabe > Art =IBISDirectory-Xml“)

- **Kodierung**: Legt die Kodierung des Zeichensatzes fest.
  - **inkl. Daten**: Wenn markiert, dann wird die gesamte Datei gelesen. Wenn nicht markiert, dann werden nur die Metadaten der Datei gelesenen, z. B. Name, Dateigröße und Datum.
  - **Daten base64 kodieren**: Der Inhalt der Datei wird base64-kodiert der XML-Ausgangsnachricht mitgegeben.

- **Daten komprimieren**: Die Daten werden Zip-komprimiert.
- **Absoluten Pfad im XML speichern**
  Wenn markiert, wird der absolute Pfad der Datei in einem gesonderten Attribut `path` gespeichert.

Zip Konfiguration

(nur bei „Datenweitergabe > Art = Zip“)

Kodierung: Legt die Kodierung des Zeichensatzes fest.

10.2.3 Dialog „Zu schreibende Datei“

(Output Connector)

In diesem Dialog geben Sie das Format der Eingangsnachrichten an und legen fest, wie diese Nachrichten ins Dateisystem geschrieben werden sollen.

Eingangsformat

Eingangsformat:

Geben Sie das Format der Eingangsnachrichten an:

- **Daten**
  Die Eingangsnachrichten sind Daten, die unverändert weitergegeben werden sollen.
- **IBISDirectory-Xml**
  Die Eingangsnachrichten haben das IBISDirectory XML-Format.
- **Zip**
  Bei dieser Option erwartet der File Connector ein Zip-Archiv als Eingangsnachricht. Der File Connector entpackt dieses Archiv und speichert die entpackten Daten im angegebenen Verzeichnis.

Zip-Konfiguration

(Bei Eingangsformat=Zip)

Kodierung: Die Auswahl legt die Kodierung der Datei- und Verzeichnisnamen fest.

Verzeichnis

- **Name**
  Absoluter Pfad zu dem Verzeichnis, das der File Connector beim Schreiben von Dateien verwenden soll. Zusätzlich können Sie dem zu erzeugenden Dateinamen ein Datum, einen Zeitstempel oder die Prozess-ID hinzufügen:
  - **Datum**

Auch die Datumsangabe hat einen Zeitstempel. Wenn Sie den Dateinamen folgendermaßen angegeben `file_%TimeStamp%%.xml` erhält die Ausgabedatei folgenden Namen: `file_12122010-151047.xml` mit „file“ als Dateiname, die erste Zahl ist das Datum im Format ddmmyyyy, die Zahl hinter dem Bindestrich gibt die Zeit in Format hhmmss an.

- **Zeitstempel**

  Der Zeitstempel ist genauso angelegt wie der Datumsstempel (siehe oben). Es wird dazu noch eine Angabe in Millisekunden hinzugefügt. Wenn Sie den Dateinamen folgendermaßen angegeben `file_%TimeStamp%ddMMyyyy-HHmmss-SSS%.xml` entsteht beispielsweise der folgende Dateiname als Name der Ausgabedatei `file_12122002-151304-027.xml`. Die dreistellige letzte Zahl gibt die Millisekunden an.

- **Prozess-ID**

  Die Prozess-ID ist die eindeutige Zahl, die für eine Workflow-Ausführung angelegt wird. Um der Ausgabedatei die Prozess-ID der Workflow-Ausführung mitzugeben, geben Sie den Dateinamen im folgenden Format an `file_PID_%ProcessId%.xml`. Es entsteht z. B. der folgende Dateiname als Name der Ausgabedatei `file_PID_18.xml`.

In Verbindung mit „Fehlende Verzeichnisse erzeugen“ können Sie z. B. tages- oder monatsbezogene Archivstrukturen für Nachrichten anlegen.

- **Erzeuge fehlende Verzeichnisse**: Erzeugt alle nicht-existierenden Verzeichnisse aus der Pfadangabe.

Datei

(nur bei Eingangsformat=Daten)

- **Name**

  Auch im Dateinamen können Sie Wildcards (*) verwenden, und Zeitstempel sowie Prozess-ID einfügen lassen, siehe Option „Verzeichnis > Name“.

- **Verwende Wildcard-Inhalt des vorher ausgeführten File oder FTP Connectors**

  Beispiel: Ein vorher ausgeführter File Connector wurde so konfiguriert, dass im Feld „Verzeichnis und Dateiname“ eine Datei „Test*.xml“ angegeben wurde. Als Ausgangsnachricht wurde vom File Connector eine Datei mit Namen „Test5.xml“ weitergeleitet. Der Wildcard-Inhalt ist in diesem Fall das Zeichen „5“. Wenn Sie im aktuellen Connector ebenfalls eine Wildcard angegeben haben, zum Beispiel „Ausgabe*.xml“, so wird an Stelle der Wildcard das Zeichen 5 eingefügt. Die Ausgabe des hier konfigurierten Connectors ist daher „Ausgabe5.xml“.

- **Verwende Dateiname von vorher ausgeführtem File Connector**

  Beispiel: ein vorher ausgeführter File Connector wurde so konfiguriert, das im Feld „Verzeichnis und Dateiname“ eine Datei „Test.xml“ angegeben wurde. Als Ausgangsnachricht wird vom File Connector eine Datei mit genau diesem Namen „Test.xml“ weitergeleitet. Dieser Dateiname wird dann auch hier verwendet.
- **Daten an die Datei anfügen**

  Die zu lesende Datei wird bei jeder Ausführung des Workflows an eine evtl. bereits existierende Datei angehängt. Dies ist z. B. sinnvoll, um Log-Daten kumulativ in einer Datei zu protokollieren. Diese Option kann nicht verwendet werden, wenn im Dateinamen Wildcards angegeben sind!
- **Eventuell bestehende Datei überschreiben**

  Wenn bereits eine Datei mit demselben Namen existiert, dann wird diese überschrieben.
- **Sicheres Schreiben**

  Diese Option stellt sicher, dass Applikationen, die auf die Datei warten, die Datei erst nach deren Fertigstellung abholen.

  Wenn markiert, dann werden die Daten zunächst in eine temporäre Datei geschrieben. Wenn dieser Vorgang abgeschlossen ist, wird die temporäre Datei entsprechend dem angegebenen Namen umbenannt. Bei größeren Dateien kann das Erstellen der temporären Datei länger dauern.

Konfiguration Modulausgang

Eingangsnachricht zusätzlich auch in den Ausgangsnachricht setzen

Wenn die Option markiert ist, dann wird die Eingabedatei an nachfolgende Module weitergeleitet. Abhängig von der Größe der Eingabedateien kann diese Option die Performance Ihres Systems beeinflussen.

→ Siehe *Modulvariablen des File Connectors (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 10.1, S. 121).*

11 FTP Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Der FTP Connector unterstützt die Transferprotokolle FTP, FTPS und SFTP.

Konnektortypen

Es gibt folgende Konfigurationsmöglichkeiten für FTP Connectoren:

- **Input Connector**
  Holt Dateien von einem FTP-Server.
- **Output Connector**
  Schreibt Dateien in ein Verzeichnis auf einem FTP-Server.

→ Siehe auch
  - *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
  - *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
  - *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

11.1 Modulvariablen

Bei der Ausführung eines FTP Connectors werden folgende Variablen gesetzt:

- **`ReadFileDate`**
  Letztes Änderungsdatum der Eingangsdatei, Format `dd.mm.yyyy hh:mm:ss`
- **`ReadFileName`**
  Name der Eingangsdatei mit Extension
- **`ReadFileSize`**
  Größe der Eingangsdatei, in Bytes
- **`ReadFileWildcardname`**
  Gibt den Inhalt der Wildcard aus, die beim Konfigurieren des FTP Connectors im Dateinamen verwendet wurde.
- **`WriteFileDate`**
  Nur beim FTP Output Connector!

Erzeugungsdatum der Ausgabedatei, Format `dd.MM.yyyy HH:mm:ss`

- **`WriteFileName`**
  Name der Ausgabedatei
- **`WriteFileWildcardname`**
  Nur beim FTP Output Connector, wenn im zu erzeugenden Dateinamen z. B. Timestamp oder Wildcard verwendet werden!
  Gibt den Inhalt der Wildcard aus, die beim Konfigurieren des FTP Output Connectors im Dateinamen verwendet wurde.

→ Siehe *Workflow-Variablen und Mappings (Workbench: Benutzer-Guide, Kap. 14, S. 365).*

11.2 Dialogbeschreibungen

Dieser Abschnitt erläutert die folgenden Themen:

11.2.1 Dialog „FTP Connector Eigenschaften“

In diesem Dialog werden die Zugangsdaten zum FTP-Server abgefragt.

Abhängig vom ausgewählten Protokoll werden andere Bedienelemente angezeigt.

Grundkonfiguration

- **Protokoll**: Wählen Sie eines der Protokolle aus:
  - **FTP** (File Transfer Protocol)
    Zwischen zwei Rechnern, die über TCP/IP verbunden sind, wird eine Verbindung über zwei Ports aufgebaut:
    - Ein Port wird zur Steuerung der Kommunikation genutzt, also zum Austausch der Befehle.
    - Über den anderen Port werden die Daten unverschlüsselt übertragen. Diese Übertragung ist nicht sicher.

- **FTPS** (FTP über SSL)
  Erweiterung von FTP:
  - Der Steuerport ist über SSL gesichert.
  - Datenaustausch ist standardmäßig unverschlüsselt. Bei dem FTP Connector können Sie auch den Datenaustausch über SSL sichern.
- **SFTP** (SSH FTP)
  Ermöglicht neben dem verschlüsselten Datenaustausch auch eine Authentifizierung von Benutzern und Servern.
  SFTP lässt sich einfacher durch NAT-Gateways hindurch verwenden als FTPS.

- **Servername**
  Domainname des FTP-Servers, z. B. ftp.example.com.
- **Port**
  - **FTP** und **FTPS**
    Standardmäßig Port 21.
  - **SFTP**
    Standardmäßig Port 22.
    Falls der Verbindungstest fehlschlägt, fragen Sie Ihren Administrator, über welchen Port der FTP-Server erreichbar ist.
    Mit dem Button „Standard“ stellen Sie nach Portänderungen den Standardwert wieder her.
- **Verbindungsmodus (nur FTP/FTPS)**
  - **Aktiv**
    Der Client öffnet der Client einen zufälligen Port und teilt dem Server diesen Port und die eigene IP-Adresse über das PORT-Kommando mit.
  - **Passiv**
    Der Client sendet ein PASV-Kommando, der Server öffnet einen Port und übermittelt diesen mitsamt IP-Adresse an den Client.
  - **Auto**
    Meist die passende Einstellung. Wählen Sie „Aktiv“ oder „Passiv“ nur dann, wenn Ihr Administrator den FTP-Server entsprechend konfiguriert hat.
- **Übertragungsart**
  Geben Sie an, ob die zu übertragenden Dateien binär sind oder ASCII-Daten enthalten.
- **Regionalschema**
  Regionalschema des Servers. Das Regionalschema bestimmt u. a. die Art des Zeichensatz und der Datumsangabe.
- **Kodierung**
  Geben Sie die Kodierung des Kontrollkanals für die Kommunikation zwischen Konnektor und FTP-Server an.

Über den Kontrollkanal werden die Befehle und dabei auch die Dateinamen als Zeichenketten übertragen. Um sicherzustellen, dass dabei z. B. Umlaute korrekt übertragen werden, ist die Definition der Kodierung notwendig, da diese von Server zu Server unterschiedlich sein kann.

Authentifizierung (nur FTPS)

Falls Sie für den FTP-Server einen eigenen Bereich mit erforderlichem Login haben, tragen Sie die Login-Daten in die entsprechenden Felder ein.

Um anonym auf den FTP-Server zugreifen können, klicken Sie auf den Button Anonymer Zugang. Dann werden der Benutzername „anonymous“ und das Passwort „anonymous@example.com“ automatisch gesetzt.

SSL Konfiguration (nur FTSP)

- **Implizites FTPS benutzen**
  Beschreibt eine ältere FTPS Version, die in der aktuellen Spezifikation nicht mehr enthalten ist. Fragen Sie den Administrator des FTPS-Servers, ob „Implizites FTPS“ vorliegt. Dieses FTPS läuft standardmäßig auf Port 990.
  Die nachfolgenden Felder gelten für das aktuelle FTPS und das „Implizite FTPS“ gleichermaßen.
- **Sicherungsart**:
  Wählen Sie ein Protokoll für die Datenübertragung aus:
  - **SSL** oder **TLS** (Secure Sockets Layer und Transport Layer Security)
    TLS 1.0 und 1.1 sind die standardisierten Weiterentwicklungen von SSL 3.0. Beide stellen FTPS-Sicherheitsmechanismen zur Absicherung der Verbindungen zur Verfügung.
  - **TLS_C**
    TLS inkl. Kompression der verschlüsselten Daten.
- **Abgesicherte Datenübertragung**
  Standardmäßig werden bei FTPS nur die Steuerbefehle verschlüsselt übertragen. Der FTP Connector kann zusätzlich Daten verschlüsselt übertragen.
  Wenn aktiviert, dann werden Daten verschlüsselt übertragen.
- **SSL Konfiguration**
  Für die Server- und/oder Client-Authentifizierung. Öffnet den *Dialog „SSL-Konfiguration“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1.1.4, S. 24)*.

SSL Konfiguration (nur SFTP)

- **SHA-1 deaktivieren**

Markieren Sie diese Option, wenn Sie ältere Dienste einsetzen, die keine SHA-1-Verschlüsselung unterstützten. Damit vermeiden Sie Fehler bei der Ausführung des Connectors und damit einen Abbruch des Workflows.

- **Server-Authentifizierung**

  Wenn nicht markiert, dann muss sich der SFTP-Server Ihres Geschäftspartners gegenüber dem FTP Connector identifizieren.

  Wenn markiert, dann gilt der angegeben SFTP-Server als vertrauenswürdig. Sie müssen den SFTP-Server zu einer `known_host` Datei hinzufügen, diese Datei hochladen und/oder den Pfad zu der Datei eingeben:

  - **known_hosts Datei hinzufügen**

    Der Button öffnet einen Dateiexplorer zum Laden der `known_hosts`-Datei mit vertrauenswürdigen Servern. Die Datei wird im FTP Connector gespeichert und wird bei jedem Verbindungsaufbau zum SFTP-Server verwendet.

  - **Pfad**

    Pfad zur Datei `known_hosts`.

    Vorteil: Änderungen in der Datei werden bei jeder Ausführung berücksichtigt.

  Wenn Sie die Datei hochgeladen und einen Pfad angegeben haben, dann wird die Pfadangabe bevorzugt.

- **Client-Authentifizierung**

  Wenn markiert, dann muss sich der Connector als Client gegenüber dem SFTP-Server identifizieren.

  Der Button „Privaten Schlüssel hinzufügen“ öffnet einen Dateiexplorer zum Laden des privaten Schlüssels bzw. Keystores.

Um den Keystore laden zu können, müssen Sie das Passwort für den privaten Schlüssel bzw. den Keystore eingeben!

Verbindungstest

- **Verbindung testen**

  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

11.2.2 Dialog „FTP Kommandos ausführen“

Nur bei Auswahl von FTP und FTPS!

In diesem Dialog können Sie einen oder mehrere Befehle angeben, die vor oder nach der Dateiübertragung ausgeführt werden sollen. Es können z. B. vor dem Übertragen Verzeichnisse angelegt oder Dateien gelöscht werden.

Die Befehle müssen der standardisierten Syntax entsprechen, siehe z. B. http://www.networksorcery.com/enp/protocol/ftp.htm

Die Befehle müssen in Großbuchstaben geschrieben werden. Bei mehreren Befehlen steht jeder Befehl in einer eigenen Zeile.

Verwenden Sie nicht die Befehle `delete`, `mkdir`, `get` und `put`. Diese Befehle erzeugen Fehler!

Bei Verzeichniswechsel:
Wenn Sie sich bei einem FTP-Server anmelden, wird danach das für Sie angelegte Home-Verzeichnis angezeigt. Dies ist üblicherweise ein Verzeichnis wie z. B. `/ftproot/group/username`. Als Benutzer „sehen" Sie allerdings nur Ihr Verzeichnis `username`. Einige FTP-Server sind so konfiguriert, dass beim Wechseln in ein hierarchisch darunter liegendes Verzeichnis absolute Pfadangaben verlangt werden. Für den Wechsel in ein Unterverzeichnis `/ftproot/group/username/test` würde der Befehl dann z. B. folgendermaßen lauten: `ftp>CWD /ftproot/group/username/test`

11.2.3 Dialog „Input-Dateinamen definieren"

(Input Connector)

Dieser Dialog bietet folgende Optionen:

Modus

- **Dateimodus**
  Pro Ausführung wird eine Datei im angegebenen Verzeichnis gelesen, wenn deren Name dem Muster entspricht, das im Feld „Datei > Name" angegeben ist. Welche Datei gelesen wird, hängt von Ihren Angaben bei „Lesereihenfolge" ab.
- **Verzeichnismodus**
  Es werden alle Dateien im angegebenen Verzeichnis gelesen, wenn deren Dateinamen dem Muster entsprechen, das im Feld „Datei > Name" angegeben ist.
  Im Verzeichnismodus wird immer eine Ausgangsnachricht im IBISDirectory-XML-Format erzeugt, d. h. der Technical Workflow startet in jedem Fall.

Der Verzeichnismodus ist nicht rekursiv, d. h. Unterverzeichnisse des angegebenen Verzeichnisses werden nicht gelesen.

Verzeichnis

Name

Absoluter Pfad zu dem Verzeichnis, aus dem der FTP Connector Dateien übertragen soll. Wenn Sie Dateien direkt aus dem Home-Verzeichnis Ihres FTP-Servers übertragen wollen, lassen Sie das Feld leer.

Datei

- **Mit Wildcard**
  Wenn Sie diese Option wählen, können Sie den vollständigen Dateinamen angeben oder im Dateinamen eine Wildcard (*) benutzen, mit `*.xml` werden dann z. B. alle XML-Dateien aus dem angegebenen Verzeichnis übertragen.
- **Mit regulärem Ausdruck**
  Wenn markiert, dann können Sie mit Hilfe eines regulären Ausdrucks ein Muster angeben, dem die Dateinamen der einzulesenden Dateien entsprechen müssen.

Reguläre Ausdrücke müssen in Schrägstriche eingeschlossen werden, z. B. `/(\d\d).(\d\d).(\d\d\d\d)/`.

Für Informationen über reguläre Ausdrücke siehe z. B. http://www.perl.com/doc/manual/html/pod/perlre.html oder http://www.regular-expressions.info/tutorial.html.

- **Name**
  Dateiname. Kann eine Wildcard oder einen regulären Ausdruck enthalten.
- **Verwende Wildcard-Inhalt des vorher ausgeführten File oder FTP Connectors**
  Diese Option ist nur aktiv, wenn Sie die Checkbox „Mit optionaler Wildcard“ markieren. Wenn Sie diese Option anklicken, dann wird der Wildcard-Inhalt verwendet, der von einem vorher ausgeführten File oder FTP Connector erzeugt wurde, z. B.
  Ein vorher ausgeführter File Connector wurde so konfiguriert, dass im Feld „Dateiname“ eine Datei „Test*.xml“ angegeben wurde. Als Ausgangsnachricht hat der File Connector eine Datei mit Namen „Test5.xml“ weitergeleitet. Der Wildcard-Inhalt ist in diesem Fall das Zeichen „5“. Wenn Sie im aktuellen Connector ebenfalls eine Wildcard angeben, z. B. „Ausgabe*.xml“, dann wird die Wildcard durch „5“ ersetzt und die Ausgabe des aktuellen Connectors ist „Ausgabe5.xml“.
- **Dateien nach dem Lesen löschen**

Wenn die Dateien nach der Übertragung gelöscht werden sollen, aktivieren Sie die Checkbox.

- **Fehler erzeugen, wenn Löschvorgang fehlschlägt**
  Wenn ein Fehler auftritt, werden die Modulausführung und der Workflow fortgesetzt. Wenn die Option aktiviert ist und ein Fehler auftritt, dann wird im Queue Manager ein Fehlereintrag für den Workflow mit diesem Modul angezeigt. Ist ein Fehlerausgang konfiguriert, wird dieser durchlaufen. Sonst wird die Ausführung des Moduls abgebrochen.
- **Max. Anzahl v. Ausführungen pro zeitgesteuertem Aufruf**
  Mit der Angabe legen Sie fest, wie oft pro auslösendem Event der Workflow maximal gestartet werden soll. Auslösendes Event ist das Erreichen eines Zeitpunktes. Dann wird der Workflow so oft gestartet, bis entweder
  - keine weiteren Daten zu holen/zu senden sind oder
  - die Anzahl erreicht wurde, welche bei „maximale Ausführungen pro Aufruf" konfiguriert wurde.

  Mit der Begrenzung der Ausführung steuern Sie die Systemauslastung der inubit Process Engine.
  Die Angabe wird im Test-Modus ignoriert, dann wird der Workflow genau einmal gestartet.

Die Option ist nur bei aktiviertem Scheduler aktiv!

Lesereihenfolge

Legen Sie Reihenfolge und Sortierung beim Lesen der Dateien fest.

Dateiprüfung

- **Abbruch des Workflows, wenn die Datei nicht existiert**
  Wenn der Zugriff auf die angegebene Datei fehlschlägt, wird die Ausführung des Workflows mit einem Fehler abgebrochen.
- **Erzeuge XML Ausgangsnachricht mit der Statusinformation darüber, ob die Datei existiert oder nicht**
  Der FTP Connector gibt eine XML Datei aus, die ein einziges Attribut `<FileExists>` enthält. Dessen Wert ist entweder „false“, wenn die angegebene Datei nicht existiert oder „true“, wenn die angegebene Datei existiert.
- **Erzeuge XML Ausgangsnachricht mit der Statusinformation darüber, dass die Datei nicht existiert, ansonsten Abbruch des Workflows**
  Wenn die Eingabedatei fehlt, gibt der FTP Connector eine XML Datei mit dem Element `FileExists=‟false‟` aus. In allen anderen Fehlerfällen wird die Workflow-Ausführung mit einem Fehler abgebrochen.

Verzeichnis Konfiguration

(Nur bei Auswahl „Verzeichnismodus)

- **Limit Dateianzahl**
  Maximale Anzahl von Dateien, die eingelesen werden.
- **Limit Gesamtgröße**
  Maximale Größe, bis zu der Dateien eingelesen werden.
- **inklusive Daten**
  Wenn markiert, dann wird der gesamte Inhalt der Datei eingelesen.
  Wenn nicht markiert, dann werden nur die Metadaten der Datei eingelesenen, z. B. Name, Dateigröße und Datum.

11.2.4 Dialog „Output-Dateinamen definieren“

(Output Connector)

Dieser Dialog bietet folgende Optionen:

Verzeichnis

- **Name**
  Verzeichnis auf dem FTP-Server, welches der FTP Connector beim Schreiben von Dateien verwenden soll.
- **Erzeuge fehlende Verzeichnisse**
  Wenn aktiviert, werden fehlende Verzeichnisse auf dem FTP-Server erzeugt.

Datei

- **Name**
  Geben Sie einen Namen für die Datei an, die der Output Connector schreibt.

Im Dateinamen können Sie einen Asterisk (*) verwenden, der an der jeweiligen Stelle einen Zähler einfügt, um das Überschreiben von Dateien zu verhindern, z. B. `/Outputdata/output*.xml`. In diesem Fall würden Dateien mit den Namen `output211.xml`, `output212.xml`, `output213.xml` usw. auf dem FTP-Server erzeugt werden.

Sie können dem Dateinamen ein Datum, einen Zeitstempel oder eine Prozess-ID hinzufügen lassen:

- **Datum**
  - Angabe ohne Datumsformat:

  `file_%TimeStamp%%.xml` erzeugt `file_06082008-151047.xml` mit

file = Dateiname

`12122002 =` Datum im Format `ddMMyyyy`

`151047=` Zeit im Format hhmmss.

- Angabe mit Datumsformat `hh`:

`File_%TimeStamp%hh%.xml` erzeugt `File_11.xml`

- Angabe mit Datumsformat `ddMM`:

`File_%TimeStamp%ddMM%.xml` erzeugt `File_2309.xml`

- **Zeitstempel**

  Der Zeitstempel enthält zusätzlich eine Angabe in Millisekunden:

  `file_%TimeStamp%ddMMyyyy-HHmmss-SSS%.xml`

  erzeugt

  `file_12122002-151304-027.xml`

- **Prozess-ID**

  `file_PID_%ProcessId%.xml`

- **Verwende Wildcard-Inhalt des vorher ausgeführten File oder FTP Connectors**

  Wenn Sie diese Option anklicken, so wird der Wildcard-Inhalt verwendet, der von einem vorher ausgeführten File oder FTP Connector erzeugt wurde.

- **Verwende Dateiname von vorher ausgeführtem File Connector**

  Wenn Sie diese Option wählen, so wird der Dateiname verwendet, der von einem vorher ausgeführten File Connector benutzt wurde. Beispiel: ein vorher ausgeführter File Connector wurde so konfiguriert das im Feld „Verzeichnis und Dateiname“ eine Datei `Test.xml` angegeben wurde. Als Ausgangsnachricht wird vom File Connector eine Datei mit genau diesem Namen `Test.xml` weitergeleitet. Dieser Dateiname wird dann auch hier verwendet. Falls im Feld „Name“ zusätzlich ein Dateiname eingegeben wird, wird dieser vom System ignoriert.

- **Daten an die Datei anfügen**

  Wenn Sie diese Option wählen, dann wird bei jeder Ausführung des FTP Connectors die Ausgangsnachricht an die bereits existierende Datei angehängt. Dies ist zum Beispiel nützlich, wenn Sie Log-Daten kumulativ in einer Datei protokollieren möchten.

Dieser Mechanismus kann nicht eingesetzt werden, wenn im Dateinamen Wildcards verwendet werden.
Es gibt FTP-Server, die diesen Mechanismus nicht unterstützen und eine Exception erzeugt!

- **Eventuell bestehende Datei überschreiben**

  Existiert eine Datei mit dem gleichen Namen, wird diese überschrieben.

- **Eingangsnachricht zusätzlich auch als Ausgangsnachricht setzen**
  Wenn diese Option aktiviert ist, wird die Eingangsnachricht an die Ausgangsnachricht angehängt.

12 HL 7 TCP/IP Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Der HL7 TCP/IP Connector baut eine Socket-Verbindung zwischen der inubit Suite 6 und einem anderen Programm auf, um HL7-Nachrichten über das TCP/IP MLLP-Protokoll (Minimal Lower Layer Protocol) zu senden oder zu empfangen.

Health Level 7 (HL7) umfasst eine Gruppe gebräuchlicher Austauschformate im Gesundheitswesen. Die inubit Suite 6 unterstützt die Versionen 2.3.1 und 2.4.

Konnektortypen

Die Funktionen des HL7 TCP/IP Connectors sind von der Konfiguration abhängig:

- **Input Connector**

  Fungiert als Client, holt HL7-Nachrichten von der angegebenen IP-Adresse und übergibt diese an das nachfolgende Modul des Workflows.

- **Input Listener Connector**

  Fungiert als Server und etabliert unter der angegebenen IP-Adresse einen Dienst, der auf HL7-Nachrichten wartet.

  Sobald eine HL7-Nachricht eintrifft, wird automatisch eine Empfangsbestätigung (ACK) versendet, die HL7-Nachricht in ein HL7-XML-Format konvertiert und an das nachfolgende Modul übergeben. Für die Konvertierung wird intern ein EDI Adapter genutzt.

  Es können mehrere Verbindungen zu dem Server von unterschiedlichen Clients gleichzeitig geöffnet werden. Die Verbindungen müssen vom Client geschlossen werden.

- **Medium Connector**

  Der Medium Connector verbindet sich als Client mit dem Server unter der angegebenen IP-Adresse, baut eine Verbindung auf, sendet eine HL7-Nachricht an den Server, erhält eine ACK vom Server und schließt nach Ablauf des Modul-Timeouts die Verbindung. Die ACK vom Server werden an das nachfolgende Modul übergeben und der Workflow wird fortgesetzt.

- **Output Connector**

  Der Connector verbindet sich als Client mit dem Server unter der angegebenen IP-Adresse, baut eine Verbindung auf, sendet eine HL7-Nachricht an den Server und schließt danach die Verbindung.

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*

- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

12.1 Dialog „HL7 TCP/IP Eigenschaften“

Dieser Dialog bietet folgende Optionen:

Konfiguration

- **Servername**
  - **Input Connector**
    IP-Adresse der Netzwerkschnittstelle, von der die HL7-Nachricht abgeholt werden soll.
  - **Input Listener Connector**
    IP-Adresse der Netzwerkschnittstelle, an welcher der HL7 TCP/IP Connector „lauschen“ soll.
  - **Medium/Output Connector**
    IP-Adresse der Netzwerkschnittstelle, an welche die HL7-Nachricht geschickt werden soll.
- **Port**
  Nummer des Ports, über den die HL7-Nachrichten ausgetauscht werden sollen. Standard ist der Port 60000. Portnummern größer als 65536 können Sie nicht nutzen.
  Mit dem „Standard“-Button stellen Sie die Standardportnummer wieder her.

Stellen Sie sicher, dass keine andere Anwendung auf den angegebenen Port zugreift und keine Firewall den Port blockiert.

- **HL7-Nachricht nach XML konvertieren**
  Deaktivieren Sie diese Option, wenn Sie die Nachrichten im Technical Workflow mit einem EDI-Adapter konvertieren möchten.

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

13 HTTP Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Ein HTTP Connector erfüllt abhängig von seiner Konfiguration folgende Aufgaben:

- **Input Listener Connector**

  Fungiert als HTTP-Server und wartet auf Requests von Clients.

  Dieser Konnektortyp kann synchron/asynchron kommunizieren:

  - **Synchrone Kommunikation**

    Der Eingang eines Request startet den Workflow hinter dem Input Listener Connector. Die Ausgangsnachricht des letzten Moduls im Workflow wird als Response an den Client zurück gesendet. Der HTTP-Status in der Response kann im Workflow mit Hilfe der Modulvariable `httpstatus` gesetzt werden.

  - **Asynchrone Kommunikation**

    Der HTTP Input Listener empfängt einen Request und sendet direkt eine HTTP-Statusmeldung an den Client zurück. Dann wird mit dem Request der Workflow gestartet. Das Ergebnis des Workflows wird nicht zurückgegeben.

  Alle Parameter und Header, die mit dem Request übergeben wurden, stehen während der Workflow-Ausführung als Modulvariablen zur Verfügung.

- **Input Connector**

  Sendet als Client GET-Requests an einen HTTP-Server, um Daten anzufordern. Parameter werden im Anfrage-Teil der URL übergeben. Der Nachrichteninhalt der Response wird als Ausgangsnachricht des Moduls an den Workflow übergeben. Bei Statuscode < 200 oder >= 300 wird ein Fehler erzeugt.

- **Medium/Output Connector**

  Diese Konnektortypen senden in der Funktion von Clients POST-Requests an einen HTTP-Server. Parameter und Eingangsnachrichten werden im Request-Body übertragen.

  Wenn die Response des HTTP-Servers ein Statuscode < 200 oder >= 300 ist, dann wird ein Fehler erzeugt. Andere Statuscodes gibt der Medium/Output Connector als Ausgangsnachricht aus.

Der HTTP Connector unterstützt die Protokolle HTTP und HTTPS.

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*

- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*
- *SSL-Verbindung und Server-Authentifizierung konfigurieren (Process Engine: Administrator- und Entwickler-Guide, Kap. 5.1, S. 55)*
- *Parameter setzen und anzeigen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 13.1, S. 146)*
- *Header setzen und anzeigen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 13.2, S. 147)*

13.1 Parameter setzen und anzeigen

Parameter setzen

Parameter sind beliebige Name/Wert-Paare und werden abhängig vom Konnektortyp wie folgt übergeben:

Konnektortyp	Methode	Parameterübergabe
Input Connector	GET	Als Parameter der Server-URL, z. B. http://localhost:8000/ibis/servlet/IBISHTTPUploadServlet/HTTPinputCon?PAR1=1234&PAR2=abc → Die Server-URL ändern Sie im *Dialog „HTTP Connector Eigenschaften“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 13.3, S. 149)*.
Medium/Output Connector	POST	Als Parameter im Request-Body, z. B. Par1=1234&Par2=abc Sie definieren diese Parameter beim Anlegen bzw. Bearbeiten eines Medium oder Output Connectors. → Siehe *HTTP Post Konfiguration (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 13, S. 151)*

Alle Parameter stehen während der Workflow-Ausführung als Modulvariablen zur Verfügung. Die Modulvariablen sind nach folgendem Muster benannt: `<Key>=<Value>`.

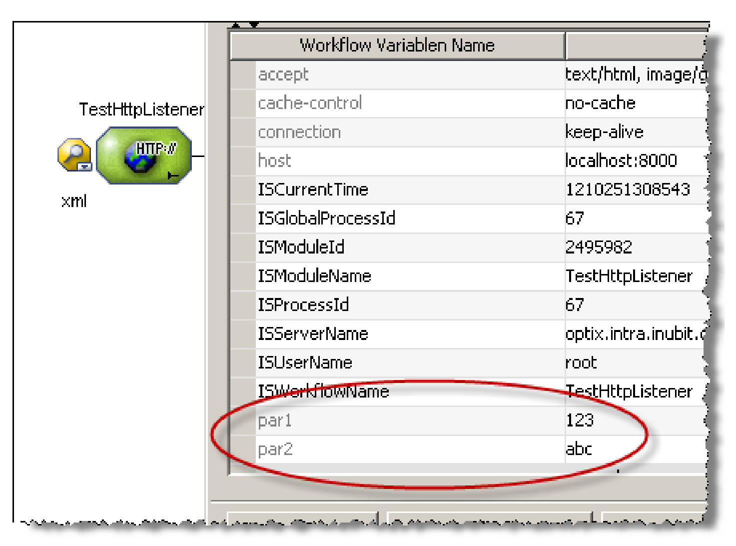
Sie können die Modulvariablen mit Hilfe des Variablen-Mappings dynamisch ändern.

→ Siehe *Workflow-Variablen und Mappings (Workbench: Benutzer-Guide, Kap. 14, S. 365).*

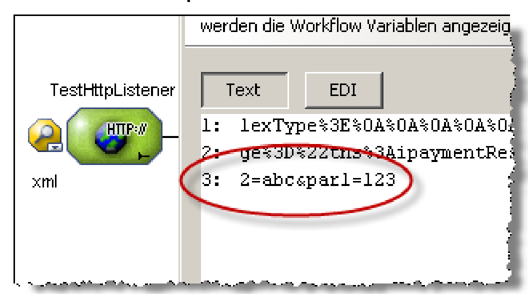
Parameter anzeigen

- **Input Connector**

  URL-Parameter, die ein HTTP Input Connector an einen HTTP Input Listener übergeben hat, können Sie am Watchpoint vor dem HTTP Input Listener anzeigen:



- **Medium/Output Connector**
  Auch die Parameter, die ein HTTP Output Connector z. B. an einen HTTP Input Listener übergeben hat, sind am Watchpoint vor dem HTTP Input Listener sichtbar:



13.2 Header setzen und anzeigen

Header statisch setzen

Sie können Request- und Response-Header beim Anlegen bzw. Bearbeiten Ihres HTTP Connectors definieren.

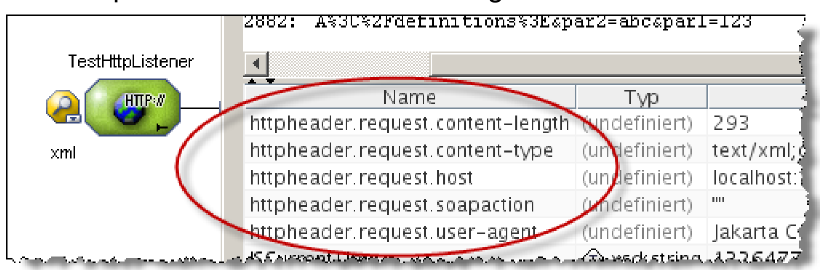
→ Siehe

- Input Listener Connector: *HTTP Header Konfiguration (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 13, S. 149)*
- Input Connector: *HTTP Header Konfiguration (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 13, S. 149)*
- Output und Medium Connector: *HTTP Post Konfiguration (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 13, S. 151)*

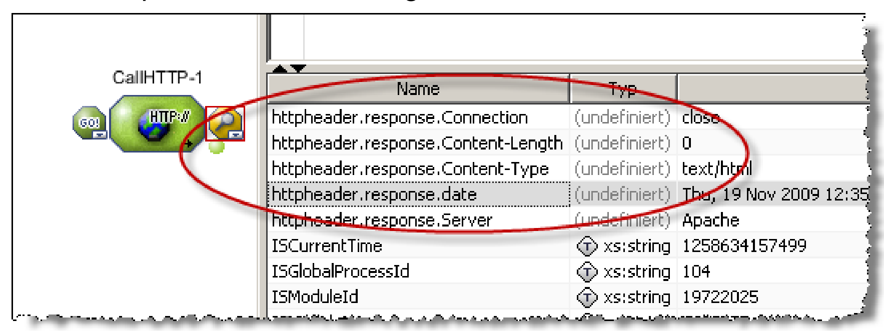
Header anzeigen

Alle Request- und Response-Header stehen nach der Workflow-Ausführung als Modulvariablen zur Verfügung:

- **Request-Header** können Sie z. B. am Watchpoint vor einem HTTP Input Listener Connector anzeigen:



- **Response-Header** können Sie z. B. am Watchpoint nach einem HTTP Output Connector anzeigen:



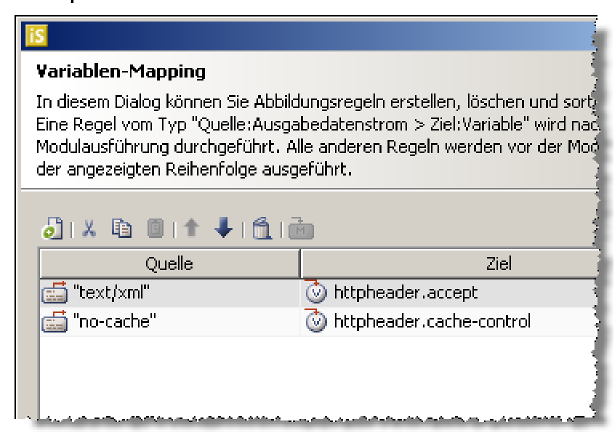
Header dynamisch setzen und überschreiben

Sie können mit Hilfe des Variablen-Mappings Header dynamisch überschreiben und neue Header setzen.

So gehen Sie vor

Definieren Sie eine Abbildungsregel nach folgendem Muster:
`Quellwert > Zielwert=httpheader.<headerName>`.

Beispiel:



→ Siehe *Workflow-Variablen und Mappings (Workbench: Benutzer-Guide, Kap. 14, S. 365).*

13.3 Dialog „HTTP Connector Eigenschaften“

Dieser Dialog bietet folgende Optionen:

Grundkonfiguration

- **Server-URL**
  - Input Listener Connector:
    URL des Input Listeners, unter welcher dieser auf Requests wartet. Die URL wird automatisch nach folgendem Muster erstellt: `protokoll:hostname:port:pfadZumServlet_NichtÄndern_:NamedesInputListeners`

Teilen Sie die Server-URL denjenigen mit, die Nachrichten zur Verarbeitung an den Input Listener senden.

  - Input Connector:
    URL des HTTP-Servers oder HTTP Input Listener Connectors, den der Input Connector aufruft.
  - Medium/Output Connector:
    URL des Servers, an den Connector seinen POST-Request sendet, z. B. Adresse eines HTTP Input Listeners.
- Button **SSL**
  (nur Input Connector)
  Zum Absichern der Kommunikation über SSL.
  → Siehe *Dialog „SSL-Konfiguration“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1.1.4, S. 24).*

Authentifizierung

- **Authentifizierung erforderlich**
  Mit dieser Option legen Sie fest, dass sich jeder Client gegenüber dem Input Listener authentifizieren muss.
  - **Verfahren**: Bei der Basic-Authentication werden Benutzername und Passwort verlangt und unverschlüsselt übermittelt.
  - **Benutzername**: Benutzername für die Authentifizierung.
  - **Passwort**: Passwort für die Authentifizierung.

HTTP Header Konfiguration

Über den Header können Informationen wie z. B. Dateigröße, HTTP-Server- und User-Agent-Kennung oder MIME-Typ zwischen Client und HTTP-Server übertragen werden.

Der Button „Header Liste“ öffnet einen Dialog, in dem Sie Name/Wert-Paare als Header definieren können.

Informationen über zulässige Header finden Sie in der HTTP-Spezifikation, siehe http://www.w3.org/Protocols/HTTP/1.1/rfc2616.pdf.

Wenn im Request bereits gleichnamige Request-Header vorhanden sind, dann werden diese Header mit den hier angegebenen Werten überschrieben!

Kodierung

- **Kodierung/Dekodierung Base64**
  Wenn markiert, dann werden die Eingangsnachrichten des Connectors vor dem Versand Base64-kodiert. Diese Option ist sinnvoll für Binärdaten.

Konvertierung

(Nur Input Listener Connector)

- **AJAX/XML-Konvertierung**
  Wenn markiert, dann werden eingehende AJAX-Requests, die von einem Task Generator erzeugt wurden, nach XML konvertiert, so wie es im Ausgangsmapping zur Verfügung steht.
  Wenn die Option nicht markiert ist, dann werden diese Requests als Byte-Strom übergeben.
- **Eingabefelder gruppieren**:
  Wenn markiert, dann werden alle Eingabefelder in der Eingangsnachricht, die denselben Index haben, bei der Ausgabe von einem Group-Element umschlossen. So ist es möglich, in einem nachfolgenden XSLT Converter über die Gruppen von Eingabefeldern mit `xsl:for-each` zu iterieren. Beispiel:

IN: Eingabefeldliste	Out: Gruppenliste

Vorname.1
Nachname.1
Straße.1
Ort.1
	`<Group>`
	`<Vorname/>`
	`<Nachname/>`
	`<Straße/>`
	`<Ort/>`
	`</Group>`

Vorname.2
Nachname.2
Straße.2
Ort.2
	`<Group>`
	`<Vorname/>`
	`<Nachname/>`
	`<Straße/>`
	`<Ort/>`
	`</Group>`

Eingabefelder haben einen Index, wenn sie durch Kopieren mit der JavaScript-Funktion `copyComponentIS()` entstanden sind. Diese Funktion ist in allen Formularen verfügbar, die mit dem Task Generator erstellt wurden.

→ Siehe *Task Generator (Data Converter) (Workbench/Process Engine: Modul-Guide, Kap. 4, S. 59).*

- **HTTP-Eingangsnachricht in XML wandeln**

  Wenn markiert, dann werden eingehende HTTP- bzw. Multipart/form-Requests (inkl. Header etc.) in XML konvertiert.

  Wenn die Option nicht markiert ist, wird der Body des HTTP-Requests ohne Konvertierung in die Ausgangsnachricht geschrieben. Multipart-MIMEs müssen dann geparst werden.

Synchrone HTTP Antwort

(Nur Input Listener Connector)

- **Aktivieren**

  Wenn aktiviert, dann empfängt der Input Listener den Request, startet den Workflow und wartet auf dessen Ergebnis. Das Ergebnis ist die Ausgangsnachricht des letzten Moduls im Workflow.

  Sobald das Ergebnis vorliegt, sendet der Input Listener es zusammen mit einer Status-Meldung an das aufrufende Modul zurück.

  Wenn die Option nicht aktiviert ist, dann empfängt der Input Listener einen Request, sendet sofort die Status-Meldung zurück und startet dann den Workflow. Es wird kein Ergebnis zurück gegeben.

- **Fehlermeldung zurückliefern**

  Wenn aktiviert, dann wird im Falle eines internen Serverfehlers (Fehlercode 500) eine detaillierte Fehlermeldung an den Client zurück geschickt.

  Wenn die Option nicht aktiviert ist, dann wird lediglich der Fehlercode 500 ohne Details zurück gesendet.

HTTP Post Konfiguration

(Nur Medium/Output Connector)

Für die Parameterübergabe:

- **Ausgabe als HTTP Post Mitteilung formatieren**

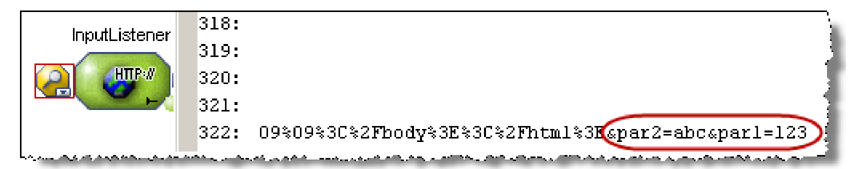
  Markieren Sie diese Option, um Parameter übergeben zu können.

  Wenn markiert, dann werden die Eingangsnachrichten im Request-Body URL-kodiert und so formatiert, als ob diese von einem Browser über ein HTML-Formular geschickt wurden, z. B.:

InputListener
../../ibis_root\root\tmp\10_Listen_08052008_174739_WThread-8-17
Datei ansehen...
Im oberen Bereich wird die Ergebnisdatei angezeigt, im unteren Bereich werden die
Text EDI
1: xml=%3C%21DOCTYPE+HTML+PUBLIC+%22-%2F%2FW3C%2F%2FDTD+HT

- **Post Feldname der Daten**
  Sie können das POST-Feld „xml“ (s. Abb. oben) durch einen beliebigen Namen ersetzen, z. B. durch „Eingangsdaten“:

- **HTTP Post Wertepaar hinzufügen**
  Öffnet einen Dialog, in dem Sie Parameter als Name/Wert-Paare definieren können. Die Parameter werden an das Ende des Request-Body gehängt, z. B.:



Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.
  (Nur Input, Medium und Output Connector)

14 inubit IS Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Der inubit IS Connector kann Daten von einem Client empfangen und damit die Verarbeitung eines Workflows anstoßen oder von einem Workflow verarbeitete Nachrichten an einen Client senden. Dabei werden neben den Nachrichten auch die Variablen übertragen.

Für die Kommunikation mit Clients verwenden inubit IS Connectoren das Thin Client Interface. Clients können z. B. Thin Clients oder weitere inubit Process Engines sein.

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

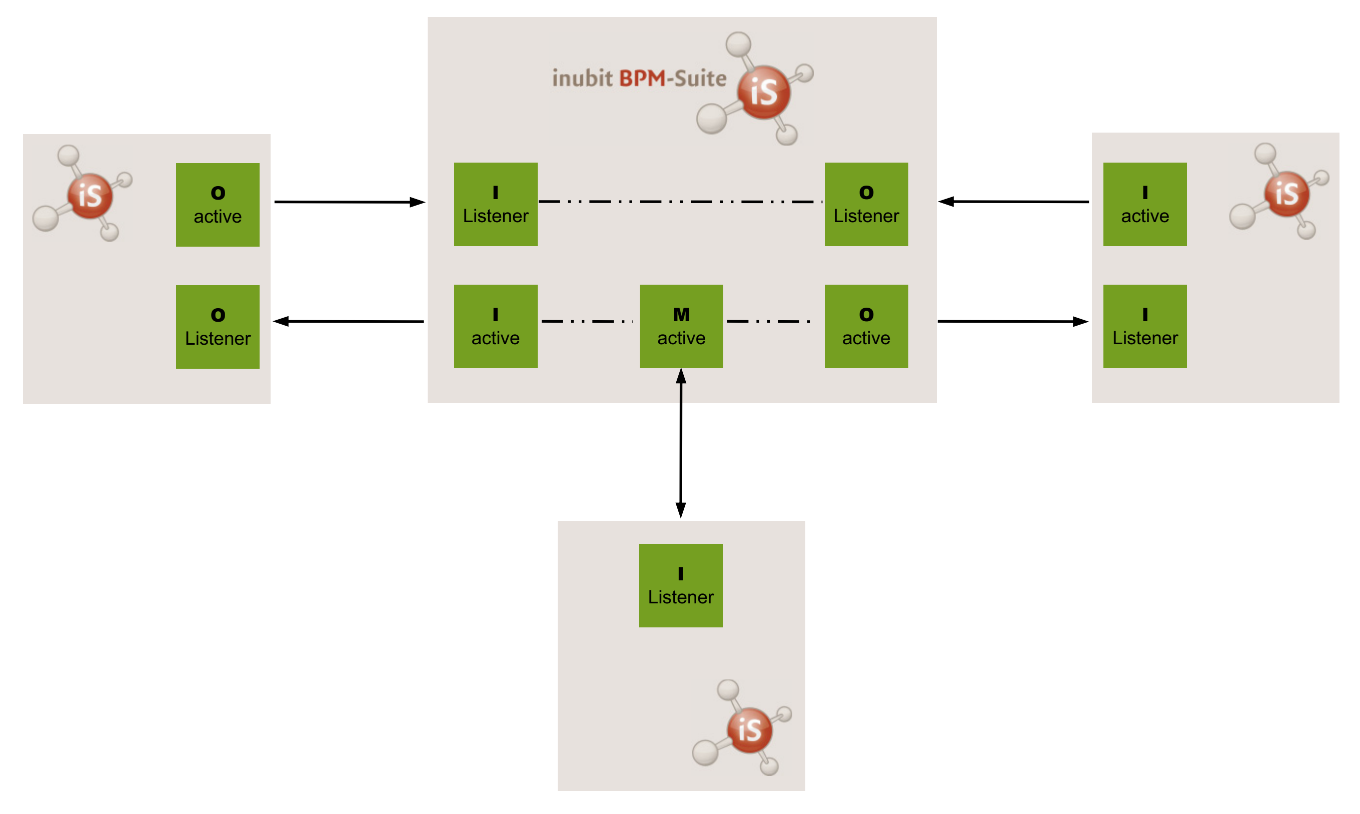
14.1 Thin Clients als Clients

Wenn sich die inubit Process Engine zentral im Internet befindet (z. B. im ASP-Betrieb) oder eine direkte Anbindung an Unternehmensanwendungen nicht möglich oder gewünscht ist, können Sie Thin Clients entwickeln, die für die Kommunikation zwischen Quell- bzw. Zielapplikationen mit der inubit Process Engine sorgen.

In diesem Szenario fragt der Thin Client Daten aus einer Quellapplikation ab, sendet sie zur Verarbeitung an einen Input Listener inubit IS Connector und stößt damit den Workflow an. Am Ende des Workflows legt ein Output Listener inubit IS Connector die Ergebnisdaten in einem Export-Verzeichnis ab. Ein anderer Thin Client kann diese Daten per Datei-Download abholen und diese an die Zielapplikation weiter geben.

14.2 inubit Process Engine als Client

In einer verteilten Umgebung mit mehreren inubit Process Engines kommunizieren diese über inubit IS Connectoren miteinander.



Die Abbildung zeigt wie, mit Hilfe einer zentralen inubit Process Engine und unter Einsatz aller Typen der inubit IS Connectoren Workflows auf verschiedenen inubit Process Engines ausgeführt werden können.

Dabei haben die inubit IS Connectoren abhängig von ihrer Konfiguration folgende Funktionen:

- Der Input inubit IS Connector vom Typ „Input Listener“ erhält von einem Output inubit IS Connector auf einer entfernten inubit Process Engine Daten zur Verarbeitung.
- Der Output Listener inubit IS Connector legt verarbeitete Daten im Export-Verzeichnis des Workflows ab. Ein Input inubit IS Connector auf einer entfernten inubit Process Engine holt diese Datei ab.
- Der Input inubit IS Connector holt Dateien von einem Output Listener inubit IS Connector auf einer entfernten inubit Process Engine per Datei-Download.
- Der Output inubit IS Connector sendet Ergebnisdaten direkt an einen Input Listener inubit IS Connector auf einer entfernten inubit Process Engine.
- Der Medium inubit IS Connector sendet Zwischenergebnisse eines Workflows an einen Input Listener inubit IS Connector auf einer entfernten inubit Process Engine. Die Ergebnisse des dort angestoßenen Workflow-Prozesses werden als Eingangsnachricht an den Nachfolger des inubit IS Medium Connectors zurück gegeben.

14.3 Dialogbeschreibungen

Dieser Abschnitt erläutert die folgenden Themen:

14.3.1 Dialog „Authentifizierung“

(Nur Input Listener Connector)

In diesem Dialog werden die Zugangsdaten zur inubit Process Engine abgefragt, mit denen sich ein Client authentifizieren muss, um Daten an den inubit IS Connector senden zu können.

Geben Sie einen beliebigen Benutzernamen und ein Passwort an.

14.3.2 Dialog „Funktion“

In diesem Dialog legen Sie die Funktion des inubit IS Connectors fest. Es ist abhängig von der Konfiguration, welche Funktion wählbar ist:

- **Input Connector**: Datei herunterladen
- **Medium Connector**: Workflow ausführen
- **Listener Output Connector**: Als Datei speichern
- **Output Connector**:
  - Workflow ausführen: Daten werden gesendet und direkt von einem Workflow verarbeitet
  - Datei hochladen: Daten werden auf den entfernten Server geladen, der Workflow wird später ausgeführt.

14.3.3 Dialog „Verbindungsdaten“

(Bei Medium und Input Connector)

Im Dialog zur Konfiguration der Konnektor-Eigenschaften geben Sie die Zugangsdaten zur entfernten inubit Process Engine an, von dem der inubit IS Connector Daten abholen soll.

Grundkonfiguration

- **Server URL**
  Ersetzen Sie `localhost` mit dem Namen oder der IP-Adresse der entfernten inubit Process Engine.
- **SSL**
  Für die Server- und/oder Client-Authentifizierung.
  Öffnet den *Dialog „SSL-Konfiguration“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1.1.4, S. 24)*.

Authentifizierung

- **Anmeldename**
  Benutzername eines Accounts auf der entfernten inubit Process Engine.
- **Passwort**
  Passwort zu dem oben angegebenen Benutzernamen.

Workflow

- **Name**
  Name des Workflows, mit dem die Verbindung erstellt werden soll.
- **System Connector**
  Name des Systemkonnektors in dem Workflow, mit dem die Verbindung erstellt werden soll.

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

14.3.4 Dialog „Input-Datei(en)“

(Bei Input Connector)

In diesem Dialog können Sie das Lesen von Dateien von einer entfernten inubit Process Engine konfigurieren.

Input-Datei(en)

- **Verzeichnis**
  Name des Verzeichnisses, aus dem der Konnektor Dateien entgegennehmen soll.
- **Name der Datei(en)**
  Name der Datei(en), die der Konnektor entgegennehmen soll. Bei der Angabe des Dateinamens können Sie Wildcards verwenden, z. B. `/Inputdata/*.xml`.
- **Wildcard-Inhalt des vorher ausgeführten File oder FTP Connectors verwenden**
  Wenn markiert, dann wird der Wildcard-Inhalt verwendet, der von einem vorher ausgeführten File oder FTP Connector erzeugt wurde.
  Beispiel: Ein vorher ausgeführter File Connector wurde so konfiguriert, das im Feld „Dateiname“ eine Datei „Test*.xml“ angegeben wurde. Als Ausgangsnachricht wurde vom File Connector eine Datei mit Namen „Test5.xml“ weitergeleitet. Der Wildcard-Inhalt ist in diesem Fall das Zeichen „5“. Wenn Sie im vorliegenden Connector ebenfalls eine Wildcard angegeben haben, zum Beispiel „Ausgabe*.xml“, so wird an Stelle der Wildcard das Zeichen 5 eingefügt.
- **Dateien nach dem Lesen löschen**
  Wenn markiert, dann werden die Dateien nach dem Lesen gelöscht.
  - **Fehler erzeugen, wenn Löschvorgang fehlschlägt**
    Wenn markiert, dann wird im Queue Manager ein Fehlereintrag für den Workflow mit diesem Modul angezeigt. Ist ein Fehlerausgang konfiguriert, wird dieser durchlaufen. Sonst wird die Ausführung des Moduls abgebrochen.
- **Maximale Anzahl von Dateien pro zeitgesteuertem Aufruf**

Numerischer Wert, z. B. 100, um die Systemauslastung der inubit Process Engine bei Aufruf von File Connectoren zu steuern.

Lesereihenfolge

Legt die Reihenfolge fest, in der die Dateien geholt werden.

14.3.5 Dialog „Output-Datei(en)“

(Bei Output Listener Connector)

In diesem Dialog konfigurieren Sie das Schreiben der Dateien.

Output-Datei(en)

- **Verzeichnis**
  Name des Verzeichnisses, aus dem der Connector Dateien entgegennehmen soll.
- **Name der Datei(en)**
  Name der Datei(en), die der Connector entgegennehmen soll. Bei der Angabe des Dateinamens können Sie Wildcards verwenden, z. B. `/Inputdata/*.xml`.
- **Wildcard-Inhalt des vorher ausgeführten File oder FTP Connectors verwenden**
  Wenn markiert, dann wird der Wildcard-Inhalt verwendet, der von einem vorher ausgeführten File oder FTP Connector erzeugt wurde.

  Beispiel: Ein vorher ausgeführter File Connector wurde so konfiguriert, das im Feld „Dateiname“ eine Datei „Test*.xml“ angegeben wurde. Als Ausgangsnachricht wurde vom File Connector eine Datei mit Namen „Test5.xml“ weitergeleitet. Der Wildcard-Inhalt ist in diesem Fall das Zeichen „5“. Wenn Sie im vorliegenden Connector ebenfalls eine Wildcard angegeben haben, zum Beispiel „Ausgabe*.xml“, so wird an Stelle der Wildcard das Zeichen 5 eingefügt.
- **Dateiname von vorher ausgeführtem File Connector verwenden**
  Wenn markiert, dann wird der Dateiname verwendet, der von einem vorher ausgeführten File Connector benutzt wurde. Beispiel: ein vorher ausgeführter File Connector wurde so konfiguriert das im Feld „Verzeichnis und Dateiname“ eine Datei „Test.xml“ angegeben wurde. Als Ausgangsnachricht wird vom File Connector eine Datei mit genau diesem Namen „Test.xml“ weitergeleitet. Dieser Dateiname wird dann auch hier verwendet.
- **Daten an die Datei anfügen**

Wenn Sie diese Option wählen, wird die angegebene Datei nicht bei jeder Ausführung des Output Connectors überschrieben sondern, es wird bei jeder Ausführung die Ausgangsnachricht an die Datei angehängt. Dies ist zum Beispiel nützlich, wenn man Log-Daten kumulativ in einer Datei protokollieren will.

- **Eventuell bestehende Datei überschreiben**

  Diese Option ist vorausgewählt. Existiert eine Datei mit dem gleichen Namen, wird diese überschrieben.

- **Eingangsnachricht zusätzlich auch als Ausgangsnachricht setzen**

  Wenn markiert, dann wird die Eingangsnachricht an nachfolgende Module weitergeleitet. Abhängig von der Größe der Eingangsnachrichten kann diese Option die Performance Ihres Systems beeinflussen.

15 ITA Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Mit einem ITA Connector können Sie Dokumente in ein ITA-Archivsystem der Firma „SER Solutions Deutschland GmbH" einfügen und abholen.

Das ITA-Archivsystem ermöglicht die strukturierte Suche nach Dokumenten, indem Dokumentenattribute in der SERaTIO Indexdatenbank abgelegt und verwaltet werden. Entsprechend unterstützt auch der ITA Connector die Vergabe von Dokumentattributen beim Einfügen der Dokumente.

Konnektortypen

Ein ITA Connector wird immer als Medium Connector eingesetzt, da er eine XML-formatierte Query als Eingangsnachricht erwartet, das Ergebnis der Query vom ITA-Archivsystem entgegennimmt und an das nachfolgende Modul im Technical Workflow übergibt.

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

15.1 Voraussetzung für den Betrieb: Treiber installieren

Für den Betrieb des ITA Connectors müssen Sie folgende `*.jar`-Dateien der Version 5.0.x bei Ihrem ITA-Provider anfordern und in Ihre inubit Suite 6-Installation kopieren:

- `blconfig.jar`
- `blmetadata.jar`
- `blueline.jar`
- `bluelineimpl.jar`
- `bluelineutil.jar`

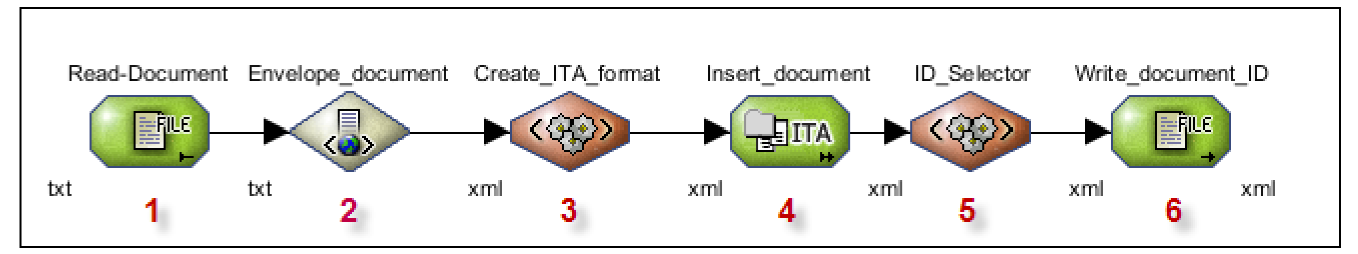
- jdom.jar

Die Dateiversion ist abhängig von der Version des genutzten ITA-Servers.

So gehen Sie vor

1. Stoppen Sie Ihre inubit Process Engine.
2. Kopieren Sie die `*.jar`-Dateien in das Verzeichnis `<iS-installdir>/server/Tomcat/webapps/ibis/WEB-INF/lib`.
3. Starten Sie Ihre inubit Process Engine.

15.2 Dokumente in ITA-Archivsystem einfügen (Insert)



Die Abbildung illustriert, wie ein Dokument in ein ITA-Archivsystem eingefügt werden kann:

1. Der File Connector liest das Dokument aus dem Dateisystem gelesen.
2. Der XML Enveloper konvertiert den Inhalt des Dokuments in ein XML-Format.
3. Im XSLT Converter wird der Inhalt des XML-formatierten Dokuments in eine XML-basierte Query konvertiert.

   In dieser Query ist die Operation definiert, die der ITA Connector ausführen soll (hier: Insert). Die Query muss konform zu dem XML-Schema des ITA Connectors sein.

Das XML Schema finden Sie im Repository unter „Global > System > Mapping Templates > ITA Archiv Connector > Query.xsd“.
In demselben Verzeichnis finden Sie auch eine Beispielquery (`queryInsert.xml`) zum Einfügen eines Dokuments.

Beim Konvertieren können Deskriptoren zu dem Dokument hinzufügt werden. Deskriptoren sind Informationen über Dokumente, wie z. B. der Dateinamen oder das Erstellungsdatum.

→ Siehe *Beispiel: Insert mit komplexen Deskriptoren (Workbench/ Process Engine: Systemkonnektor-Guide, Kap. 15, S. 163)*.

4. Der ITA Connector erhält die XML-basierte Query als Eingangsnachricht und sendet diese an das ITA-Archivsystem.

   Das ITA-Archivsystem archiviert das Dokument, fügt die Dokumentenattribute in die Indexdatenbank ein und gibt eine Kopie des Dokuments und die Dokument-ID an den ITA Connector zurück.

   Der ITA Connector gibt beides als Ausgangsnachricht im XML-Format aus.
5. Der XSLT Converter extrahiert die Dokument-ID und übergibt diese an den File Connector.
6. Der File Connector schreibt die Dokument-ID in das Dateisystem.

Beispiel: Einfaches Insert

Das folgende Listing finden Sie im Repository unter „Global > System > Mapping Templates > ITA Archiv Connector > QueryInsert.xml“:

```
<?xml version="1.0" encoding="UTF-8"?>
<Query type="insert">
  <Document maskID="4711"
      date="2009-03-13T11:45:00">
    <Representations>
      <Representation description="Std-rep"
        type="ASCII">
        <Part>
          <Data>MyDocument</Data>
        </Part>
      </Representation>
    </Representations>
  </Document>
</Query>
```

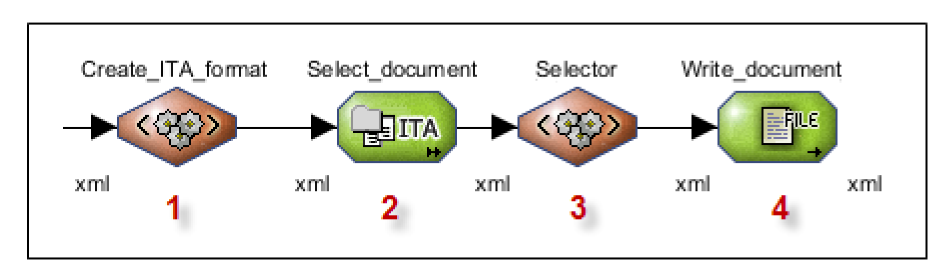
Beispiel: Insert mit komplexen Deskriptoren

Das folgende Listing zeigt, wie Sie beim Einfügen eines Dokumentes Deskriptoren hinzufügen:

```
<?xml version="1.0" encoding="UTF-8"?>
<Query type="insert">
    <Document maskID="4711">
    <Descriptors>
      <Descriptor id="1234">
        <Value type="String">mydoc.txt</Value>
      </Descriptor>
```

```
        <Descriptor id="2345">
          <Value type="String">Head</Value>
        </Descriptor>
        <Descriptor id="3456">
          <Value type="Integer">123</Value>
        </Descriptor>
        <Descriptor id="4567">
          <Value type="Date">2006-09-15</Value>
        </Descriptor>
      </Descriptors>
      <Representations>
        <Representation type="ASCII"
          description="Invoice">
          <Part>
            <Data>Daten</Data>
          </Part>
        </Representation>
      </Representations>
     </Document>
</Query>
```

15.3 Dokumente aus ITA-Archivsystem abholen (Select)



Die Abbildung illustriert, wie ein Dokument aus dem ITA-Archivsystem gelesen werden kann:

1. Der XSLT Converter erstellt eine XML-basierte Query, in der die Operation definiert ist, die der ITA Connector ausführen soll (hier: Select). Zusätzlich muss die Dokument-ID angegeben sein.

   Die Query muss konform zu dem XML-Schema des ITA Connectors sein.

Das XML Schema finden Sie im Repository unter „Global > System > Mapping Templates > ITA Archiv Connector > Query.xsd“.

2. Der ITA Connector erhält die XML-basierte Query als Eingangsnachricht, sendet diese an das ITA-Archivsystem, erhält vom ITA-Archivsystem das Dokument und gibt es als XML-formatierte Nachricht aus.
3. Der XSLT Converter extrahiert das Dokument aus der XML-Nachricht.
4. Der File Connector schreibt das Dokument in das Dateisystem.

Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<Query type="select">
  <Document maskID="1234567"/>
</Query>
```

15.4 Dialog „ITA Archiv Connector“

In diesem Dialog haben Sie folgende Optionen:

Grundkonfiguration

- **Seratio-Servername/Port**
  Domain oder IP-Nummer des Seratio Indexservers. Die Standardportnummer ist 2005. Der Button „Standard“ stellt die Standardportnummer wieder her.
- **Archiv-Servername/Port**
  Domain oder IP-Nummer des ITA-Archivservers. Häufig sind die IP-Nummern des Seratio-Indexservers und des ITA-Archivservers identisch. Die Standardportnummer ist 2000. Der Button „Standard“ stellt die Standardportnummer wieder her.

Authentifizierung

- **Systemdatenbankname/Standard**
  Name der Systemdatenbank. Standardname ist „SYSTEM“. Der Button „Standard“ stellt den Standardnamen wieder her.
- **Systemname**
  Beachten Sie, das Groß- und Kleinschreibung unterschieden wird.
- **Login/Passwort**
  Zugangsdaten zur Systemdatenbank.

Teilbestand

- **Datenbankname**
  Üblicherweise sind Datenbanksysteme in Teilbestände unterteilt. Geben Sie hier den Datenbanknamen an.

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

16 JAAS Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

JAAS (Java Authentication and Authorization Service) ist eine Java-Schnittstelle, die Klassen zur Authentifizierung und Autorisierung der aktuell angemeldeten Benutzer für die Betriebssysteme NT, Solaris und Unix bietet. Mit der Authentifizierung und Autorisierung wird überprüft, ob einem Benutzer der Zugang erlaubt wird (Authentifizierung) und welche Rechte er hat (Autorisierung), wenn der Zugang erlaubt wurde.

Der JAAS Connector bietet einen generischen Container für unterschiedliche Authentifizierungsklassen. Standardmäßig bietet der JAAS Connector folgende Funktionalitäten:

- Benutzerkonten in gängigen Namens- und Verzeichnisdiensten wie z. B. LDAP oder NIS mit dem JNDI-Modul (Java Naming and Directory Interface) überprüfen
- Benutzerkonten auf Windows-Betriebssystemen authentifizieren

  Um beliebige Benutzerkonten auf Windows Betriebssystemen validieren zu können, wurde eine bei sourceforge spezifizierte Klasse benutzt. Die Klasse stützt sich auf das Authentifizierungsprotokoll NTLM (NT LAN Manager), das ein proprietäres Microsoft Protokoll ist. Dies gilt für alle Windows-Systeme, die über einen NT-Domainkontroller verfügen. Auch Benutzerkonten des Verzeichnisdienstes Microsoft Active Directory können so authentifiziert werden.

Zusätzlich können Sie beliebige selbst-entwickelte Module laden und zur Authentifizierung von Benutzerkonten nutzen.

Siehe auch

- JAAS Reference Guide:
  http://download.oracle.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html
- Dokumentation der JAAS Login Module:
  http://jaaslounge.sourceforge.net/loginmodules.html
- JNDI-Überblick:
  http://www.oracle.com/technetwork/java/overview-142035.html

Konnektortypen

Der JAAS Connector wird als Medium Connector genutzt:

Als Eingangsnachricht erwartet der JAAS Connector die zu validierenden Benutzerkontodaten (Benutzername/Passwort). Die Benutzerkontodaten müssen unverschlüsselt übergeben werden.

Der JAAS Connector übergibt die Benutzerkontodaten dem konfigurierten System, welches die Korrektheit der Login-Daten überprüft und das Ergebnis an den JAAS Connector zurückgibt.

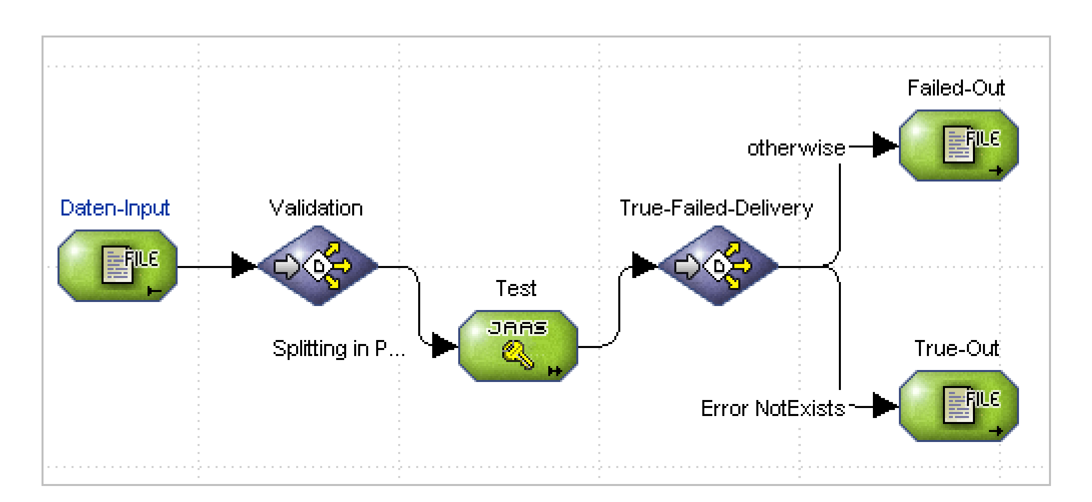
Wenn die Validierung erfolgreich ist, gibt der JAAS Connector seine Eingangsnachricht unverändert aus.

Wenn die Validierung scheitert, dann gibt der JAAS Connector eine XML-basierte Fehlernachricht aus. Die Fehlernachricht enthält u. a. einen Fehlertext mit dem Grund für die gescheiterte Validierung. Zusätzlich wird die Eingangsnachricht an die Fehlernachricht angehängt. Wenn die Eingangsnachricht nicht als XML-Datei vorlag, wird sie als Anhang der Fehlernachricht und base64-kodiert ausgegeben.

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

16.1 Beispielszenario: Liste von Benutzerkonten überprüfen



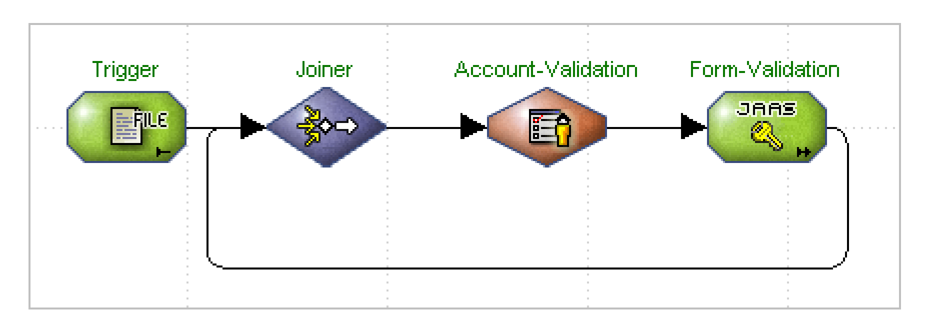
Ein Programm sammelt regelmäßig Benutzerkontodaten (Benutzername/Passwort) und fasst diese in einer XML-Datei zusammen. Der Input File Connector „Daten-Input“ übergibt diese Datei an den Demultiplexer „Validation“.

Der Demultiplexer splittet die XML-Datei, erstellt pro Login/Passwort-Paar eine neue XML-Datei und übergibt diese dem JAAS Connector „Test“.

Der JAAS Connector validiert jedes Login/Passwort-Paar. Wenn die Validierung scheitert, wird ein Error-Element hinzugefügt. Die Ausgabedatei wird dem Demultiplexer „True-Failed-Delivery“ übergeben.

Der Demultiplexer ist so konfiguriert, dass alle Dateien mit einem Error-Element, bei denen die Validierung gescheitert ist, an den Output File Connector „Fail-Out“ übergeben werden und alle anderen an den Output File Connector „True-Out“.

16.2 Beispielszenario: Benutzerdaten über Task-Formulare validieren



Der abgebildete Workflow validiert Login-Daten, die in eine Formular-Task eingegeben wurden.

Der Workflow wird gestartet durch den Input File Connector „Trigger“.

Der Task Generator „Account-Validation“ erzeugt eine Formular-Task, die in der Taskliste angezeigt wird. Diese Formular-Task besteht aus Eingabefeldern für Benutzername und Passwort sowie einem Submit-Button.

Nach dem Klick auf Submit-Button werden die eingegebenen Daten an den JAAS Connector „Form-Validation“ übergeben.

Wenn die Formular-Task erneut aufgerufen wird, dann wird wieder das Formular angezeigt. Falls die eingegebenen Daten falsch waren, wird eine Fehlermeldung angezeigt.

16.3 JAAS Connector erweitern

Sie können die Funktionalität des JAAS Connectors durch selbst-entwickelte Authentifizierungsmodule erweitern. Dazu müssen Sie die jar-Datei des Authentifizierungsmoduls in die inubit Process Engine hochladen.

So gehen Sie vor

1. Laden Sie die jar-Datei in die inubit Process Engine.
   → Siehe *Plug-ins installieren (Process Engine: Administrator- und Entwickler-Guide, Kap. 20.2.1, S. 283).*
2. Erstellen Sie einen JAAS Connector.
3. Geben Sie im *Dialog „JAAS Connector Eigenschaften“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 16.4, S. 170)* im Feld „Login-Modul“ den Klassennamen ein.
4. Geben Sie die Parameter Ihres Moduls und deren Werte in der Tabelle „Login-Modul Optionen“ ein.
5. Klicken Sie auf „Fertig stellen“, um den Dialog zu schließen.

16.4 Dialog „JAAS Connector Eigenschaften“

In diesem Dialog geben Sie an, wie die Benutzerkontodaten dem Konnektor zur Verfügung stehen und gegen welches System diese validiert werden sollen.

Login Modul

Login-Modul

- `com.sun.security.auth.module.JndiLoginModule`
  Zum Validieren von Benutzerkontodaten gegen einen LDAP- oder NIS Verzeichnisdienst.
- `org.jaaslounge.ntlm.NtlmLoginModule`
  Zum Validieren von Benutzerkontodaten gegen einen NT-Domainkontroller oder gegen den Microsoft Verzeichnisdienst Active Directory auf einem Windows Betriebssystem.

Benutzername

Geben Sie an, wie der Benutzername dem JAAS Connector übergeben werden soll.

- **D**

Der Benutzername stammt aus einem Element in der Eingangsnachricht. Das Element wird über einen XPath-Ausdruck ausgewählt.

Um das Element auszuwählen, klicken Sie auf „XML Element auswählen“, laden eine Beispielnachricht und navigieren zu dem Element. Der XPath-Ausdruck wird danach im Feld „Benutzername“ angezeigt.

Wenn der XPath mehrfach in einem XML-Dokument vorkommen, dann wird nur das erste Vorkommen des XPath ausgewertet, nachfolgende Vorkommen werden ignoriert.

- **V**

  Der Benutzername wird aus einer Workflow-Variablen gelesen. Dabei haben Sie zwei Optionen:

  - Geben Sie im Textfeld „Benutzername“ die Variable `ISUserName` an.
  - Verwenden Sie eine selbst definierte Variable.

    → Siehe *Workflow-Variablen und Mappings (Workbench: Benutzer-Guide, Kap. 14, S. 365).*

- **Fest**

  Die Zeichenkette, die im Textfeld „Benutzername“ angegeben ist, wird als Benutzername verwendet.

Nur mit der Option „Fest“ können Sie die Verbindung testen!

Passwort

Die Übernahme des Passworts funktioniert genauso wie die Übernahme des Benutzernamens:

- **D**: Zum Übernehmen des Passwort aus XML-basierten Eingangsnachrichten.
- **V**: Liest das Passwort aus einer Workflow-Variablen.

  Sie müssen eine Workflow-Variable anlegen, da es standardmäßig keine Workflow-Variable für das Passwort gibt.
- **F**: Zum Testen der Verbindung.

Login-Modul Optionen

Mit den Login-Modul Optionen werden die Werte gesetzt, die zur Verbindung mit dem System benötigt werden, auf dem das angegebene Benutzerkonto validiert werden soll.

Die Tabelle enthält für die standardmäßig mitgelieferten Login-Module bereits vorbelegte Einträge. Wenn Sie ein eigenentwickeltes Login-Modul benutzen, müssen Sie die Parameter in der Tabelle selbst anlegen.

Zum Bearbeiten der Tabelle verwenden Sie folgende Buttons:

Neue Zeile hinzufügen

Markierte Zeile löschen

Markierte Zeile eine Position nach oben verschieben.

Markierte Zeile eine Position nach unten verschieben.

Grundsätzlich ist die Reihenfolge der Login-Modul-Optionen für die Abarbeitung nicht wichtig, sondern verbessert die Übersichtlichkeit.

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

Der Verbindungstest kann nur ausgeführt werden, wenn Sie Benutzername und Passwort mit der Option „Fest“ eingegeben haben, weil zum Testzeitpunkt keine Benutzerkontodaten aus Eingangsnachrichten oder Variablen zur Verfügung stehen!

17 Java Reflection Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Mit einem Java Reflection Connector können Sie dynamisch zur Laufzeit auf Java-Objekte zugreifen und diese manipulieren.

Der Java Reflection Connector greift auf die Java Reflection-API zu, welches z. B. folgendes ermöglicht:

- Eigenschaften einer Klasse bestimmen
- Objekte erzeugen
- Methoden aufrufen

  Der Java Reflection Connector unterstützt Aufrufe von statischen und nicht-statischen Methoden auf einem Java-Objekt. Es können Parameter mit primitiven (z. B. int, long, char) und komplexen (beliebiges Java-Objekt) Java-Datentypen gesetzt werden.

Mit dem Java Reflection Connector können keine Java-Algorithmen (Schleifen, Verzweigungen, Ablauflogik) ausgeführt werden. inubit-interne Klassen wie z. B. `ibis.jar` und damit auch Klassen wie `Misc` und `Formatter` können nicht genutzt werden!
Zur Ausführung von komplexem Java-Code benutzen Sie das JavaScript-Utility.

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

17.1 Eingangsnachricht erstellen

Als Eingangsnachricht benötigt der Java Reflection Connector eine XML-Datei mit folgendem Inhalt:

- Klassenname des zu erstellenden Objekts

- Parameter für den Konstruktor (Typen und Werte)
- Name der aufzurufenden Methode
- Parameter der Methode (Typen und Werte)

Die Ausgangsnachricht hat dieselbe Struktur wie die Eingangsnachricht. Wenn gewünscht, kann die Ausgangsnachricht die Eigenschaften der Reflection-Klasse enthalten.

Erstellen Sie die serialisierte XML-Struktur ausschließlich wie im Folgenden beschrieben mit Hilfe des Java Reflection Explorers! Eine manuell erstellte XML-Struktur wird unter Umständen vom Java Reflection Connector nicht interpretiert und führt zu einem Fehler im Modul!

So gehen Sie vor

1. Erstellen Sie einen XSLT Converter.
2. Öffnen Sie im Bereich „XML-Zieldatei" das -Menü.
3. Wählen Sie „Öffnen von > Java Reflection Explorer". Der Assistent öffnet sich.

Lassen Sie sich von dem Assistenten durch die Erstellung der Nachricht leiten.

→ Siehe auch

- *Eigenen Methodenaufruf erzeugen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 17.2, S. 174)*
- *Beispiel-Methodenaufruf (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 17.3, S. 176)*

17.2 Eigenen Methodenaufruf erzeugen

Voraussetzungen

Sie müssen die `jar`-Datei in die inubit Process Engine laden, welche die Klasse enthält, deren Methoden über den Java Reflection Connector aufgerufen werden sollen.

→ Siehe *Plug-ins installieren (Process Engine: Administrator- und Entwickler-Guide, Kap. 20.2.1, S. 283).*

So gehen Sie vor

1. Öffnen Sie den Java Reflection Explorer.

   → Siehe *Eingangsnachricht erstellen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 17.1, S. 173).*

2. Wählen Sie im ersten Dialog die Option „Klasse/Methode definieren“ aus.
3. Klicken Sie auf „Weiter“. Der nächste Dialog wird angezeigt.
4. Markieren Sie die Klasse, auf welcher der Methodenaufruf erfolgen soll.

   Wenn die gewünschte Klasse nicht in der Liste angezeigt wird, dann geben Sie deren vollständigen Namen (z. B. java.lang.StringBuffer) in das Textfeld ein.
5. Klicken Sie auf „Weiter“. Es wird versucht, die entsprechende Klasse aus dem Classpath zu laden.

   Wenn die Klasse geladen werden konnte, öffnet sich der nächste Dialog.
6. **Constructor angeben**

   Wenn alle aufzurufenden Methoden statisch sind, dann entfällt die Auswahl eines geeigneten Constructors der Klasse. Falls mindestens ein Methodenaufruf nicht-statisch ist, dann müssen Sie einen Klassen-Constructor angeben.
   a. Wählen Sie im Baum den Knoten „Constructor“ aus.
   b. Öffnen Sie das Kontextmenü und wählen Sie „Constructor hinzufügen“.
   c. Wählen Sie einen passenden Constructor aus.
   d. Klicken Sie auf „OK“. Der Dialog schließt sich.

   Der Constructor wird eingefügt. Unterhalb des Constructorknotens werden die Parameter angezeigt.
7. **Methode einfügen**
   a. Wählen Sie im Baum den Knoten „Methods“ aus.
   b. Öffnen Sie das Kontextmenü und wählen Sie „Methode hinzufügen“. Ein Dialog öffnet sich.
   c. Wählen Sie alle Methoden, die aufgerufen werden sollen.
   d. Klicken Sie auf „OK“, um zurück in die Baumansicht zu gelangen.

   Die gewählten Methoden wurden hinzugefügt und alle Parameter unterhalb der Methodenknoten gesetzt.
8. **Parameterwerte setzen**

   Um einen vollständigen Methodenaufruf absetzen zu können, müssen Sie die Parameter mit konkreten Werten füllen.
   a. Markieren Sie einen Parameter.
   b. Öffnen Sie das Kontextmenü und wählen Sie „Wert setzen“. Ein Dialog öffnet sich. In diesem Dialog wird der Wert für den gewählten Parameter gesetzt.

      Zur Unterstützung wird der Datentyp des Parameters angezeigt. Wenn es sich bei dem Parameter um ein Java-Objekt mit einem komplexen Datentyp handelt, dann muss zunächst ein passender Constructor zum Erzeugen dieses Java-Objektes ausgewählt werden.

Verfahren Sie wie im Schritt „Constructor angeben“ beschrieben.

9. Wenn alle Einstellungen vollständig sind, beenden Sie den Assistenten. Die Struktur der Eingangsnachricht wird erzeugt und angezeigt.
10. Ziehen Sie die Struktur in den Bereich „XML-Ziel“ auf das `<xsl:template>`-Element, um das XSLT-Stylesheet zu erstellen.

    → Für Hinweise über die Verwendung des XSLT Converters siehe *Technical Workflow erstellen 3: CSV-XML konvertieren nach openTRANS (Tutorials, Kap. 6.5, S. 116)*.
11. Sie können mit XSLT
    - die Parameterwerte zur Laufzeit anpassen und überschreiben lassen
    - dynamische Java Reflection-Aufrufe realisieren

17.3 Beispiel-Methodenaufruf

Java Code

Das mitgelieferte Beispiel „Calling a method with simple parameters on a class instance“ setzt folgenden Java-Code um:

```
String testString = "This is a test string to be
parsed and manipulated.";
String resultString = testString.subString( 1,
testString.length() );
```

→ Siehe

- *Listing (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 17, S. 177)*
- *Ein- und Ausgangsnachrichten-Struktur (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 17, S. 177)*
- *Eingangsnachricht erstellen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 17.1, S. 173)*

Ergebnis

Die Ausgangsnachricht enthält den Wert der Variablen `resultString`: „his is a test string to be parsed and manipulated.“

Listing Methode mit einfachen Parametern auf einer Klasseninstanz aufrufen:

Listing

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <reflection>
3   <!-- first declare class instance -->
4   <object name="testString">
5     <java.lang.String>
6       This is a test string to be parsed and manipulated.
7     </java.lang.String>
8   </object>
9   <!-- in this example
  java.lang.String.substring(1,$testString.length()) is called -->
10  <!-- the result is stored in the variable resultString -->
11  <length object="$testString" result="stringLength"/>
12  <substring object="$testString" result="resultString">
13    <param type="int">
14      <int>1</int>
15    </param>
16    <param type="int" object="$stringLength" name=""/>
17  </substring>
18  <!-- so that the result can be output afterwards or reused in another
  method call -->
19  <result value-of="$resultString"/>
20 </reflection>
```

/Listing

Ein- und Ausgangsnachrichten-Struktur

1. **Root-Element**

   Das Root-Elemente für alle Java Reflection-Eingangsnachrichten muss `<reflection>` heißen.

2. **Constructor-Aufruf**

   ```
   String testString = "This is a test string to be parsed and manipulated.";
   ```

   Zuerst wird die Variable `testString` angelegt und mit dem Wert „This is a test string to be parsed and manipulated." gefüllt.

   ```
   <object name="testString">
   <java.lang.String>This is a test string to be parsed and manipulated.</java.lang.String>
   </object>
   ```

Das Element <object> gibt an, dass es sich um die Definition einer Klasseninstanz handelt. Dabei wird die Erzeugung eines Java-Objektes mittels des entsprechenden Constructor-Aufrufs abgebildet (Kindelement von <object>).

Das Attribut name="testString" gibt der Klasseninstanz einen Referenznamen, so dass diese Klasseninstanz an anderer Stelle im XML anhand des Referenznamens wiederverwendet werden kann. Als Kindelement von <object> wird das Objekt selbst als serialisierte XML-Struktur dargestellt.

3. **Methodenaufrufe**

```
String resultString = testString.subString( 1,
testString.length() );
```

Der Java-Code beinhaltet zwei Methodenaufrufe und muss daher entsprechend zerlegt werden. Der Methodenaufruf testString.length() entspricht der folgenden XML-Struktur:

```
<length object="$testString"
result="stringLength"/>
```

- Elementname <length>:
  Entspricht dem Methodennamen, der auf der Klasseninstanz aufgerufen werden soll.
- Attribut object="$testString":
  Gibt die Klasseninstanz an. Der Referenzname muss mit einem $-Zeichen beginnen, weil es sich um eine Referenz handelt.
  Bei statischen Methoden kann der Wert des Attributs object="" der Name der zu Grunde liegenden Klasse sein, z. B. object="java.lang.String".
- Attribut result="stringLength":
  Damit wird der Rückgabewert der aufgerufenen Methode in der Variablen mit dem Referenznamen "stringLength" hinterlegt. Auf diese Variable kann, äquivalent zur Klasseninstanz, durch $stringLength an anderer Stelle im XML zugegriffen werden.

Der Methodenaufruf testString.subString() wird durch die folgende XML-Struktur aufgerufen:

```
<substring object="$testString"
 result="resultString">
 <param type="int">
   <int>1</int>
 </param>
 <param type="int" object="$stringLength"
   name=""/>
</substring>
```

- Elementname <substring>:
  Entspricht wieder dem Methodennamen, der auf der Klasseninstanz aufgerufen werden soll. Als Klasseninstanz wird an dieser Stelle ebenfalls testString verwendet.

- Attribut `<result=“resultString“>`:
  Die Variable enthält das Ergebnis des Methodenaufrufs.
- Elementname `<param>`:
  Der Methode werden zusätzlich zwei Parameter übergeben.
- Attribut `type="int"`:
  Gibt den Parametertyp an (hier: int). Der Wert des Parameters folgt als Kindelement:
  - Beim ersten Parameter ist der Wert des Parameters „1“.
  - Beim zweiten Parameter wurde kein konkreter Parameterwert angegeben. Stattdessen wird der Wert der Variablen `$stringLength` verwendet,
- `object="$stringLength"`:
  Enthält das Ergebnis des vorhergehenden Methodenaufrufs. Die Variable wird auch hier wieder mit $-Zeichen und Variablenname referenziert.
- Attribut `name=""`:
  Name des Parameters, um an anderer Stelle wieder auf diesen Parameter im XML zugreifen zu können. Optional.

Achten Sie bei der Vergabe von Referenznamen darauf, dass diese innerhalb des XML eindeutig sind. Wenn ein Referenzname mehrfach verwendet wird, dann wird Inhalt der Variablen bei der nächsten Deklaration überschrieben. Referenznamen werden global deklariert und sind damit innerhalb des gesamten `<reflection>`-Elements verfügbar.

4. **Ergebnis und XML-Ausgangsnachricht**

```
<result value-of="$resultString"/>
```

Über das Element `<result>` wird die Ausgabe des Java Reflection Connectors gesteuert. Die Struktur der Ausgangsnachricht entspricht der XML-Eingangsnachricht. Der einzige Unterschied ist, dass alle `<result>`-Elemente zur Laufzeit aufgelöst werden und der entsprechende Inhalt der referenzierten Variablen eingetragen wird.

Das Ergebnis des oben genannten Beispiels sieht so aus:

```
<result value-of="$resultString">
 <string>
   his is a test string to be parsed and
   manipulated.
 </string>
</result>
```

17.4 Attributliste

Attribut	Zulässig auf	Erläuterung
@name	`<object>-Element` `<param>-Element`	Name zur Referenzierung des Wertes der Klasse (Klasseninstanz) bzw. des Methodenparameters (Parameterwert)
@object	`Methoden-Element` `<param>-Element`	Name der Klasse (z.B. java.lang.String) bzw. Referenz ($-Zeichen + Name der Referenzierung) auf das die Methode bzw. der Parameter angewendet werden soll
@result	`Methoden-Element`	Name zur Referenzierung des Rückgabewertes einer Methode
@type	`<param>-Element`	Angabe der Klasse des Typs des Methodenparameters (z.B. java.lang.String)

17.5 Dialog „Java Reflection Connector Eigenschaften“

In diesem Dialog haben Sie folgende Optionen:

Ausgabe-Optionen

Nur Resultate

Wenn markiert, dann enthält die Ausgangsnachricht nur das XML-Element `<result>`. Die Eingangsnachricht wird nicht ausgegeben.

Dient der Übersichtlichkeit und Performance bei großen XML-Eingangsnachrichten.

18 JCA Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Mit dem JCA Connector (J2EE Connector Architecture) können Sie beliebige Applikationen in die inubit Suite 6 integrieren.

→ Siehe auch

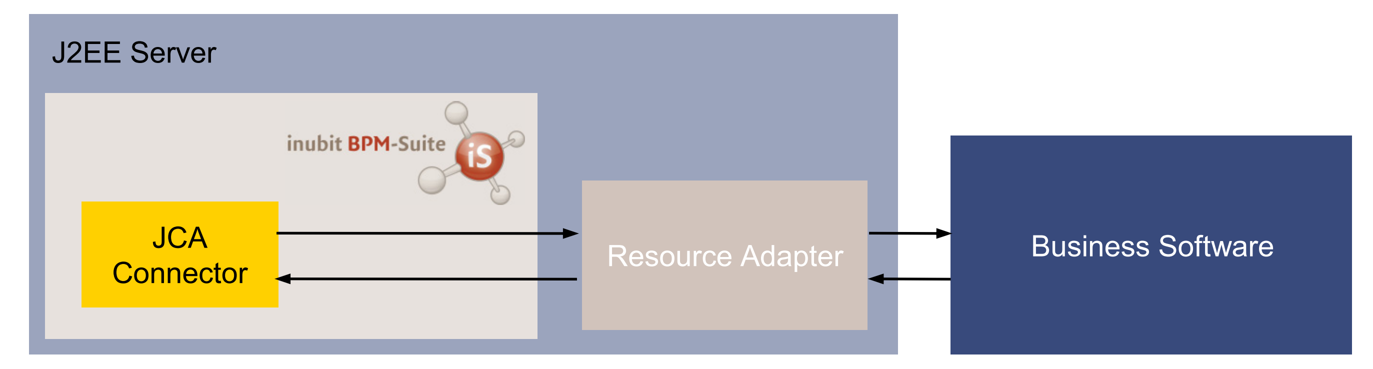
- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

18.1 Funktionsprinzip

Voraussetzungen

- Die inubit Process Engine muss im JBoss betrieben werden.
- Für die Kommunikation mit der entfernten Applikation benötigen Sie einen Resource-Adapter. Diesen erhalten Sie als `rar`-Datei von dem Hersteller der Applikation, die Sie integrieren möchten.
- Der Resource Adapter muss auf denselben JBoss deployt sein wie die inubit Process Engine.
- Sie müssen die JCA-Library (`jar`-Datei) auf die inubit Process Engine laden, um den JCA Connector Explorerassistenten nutzen zu können. Die JCA-Library ist in der `rar`-Datei des Resource-Adapters enthalten.

  → Siehe *Plug-ins installieren (Process Engine: Administrator- und Entwickler-Guide, Kap. 20.2.1, S. 283).*



Der JCA Connector überträgt Parameterwerte mit Hilfe des Resource Adapter zur Verarbeitung an die konfigurierte entfernte Applikation:

1. Ein JCA Connector erwartet eine Eingangsnachricht mit den Parameterwerten. Die Eingangsnachricht erstellen Sie in einem XSLT Converter mit Hilfe des JCA Connector Explorers.
2. Der JCA Connector verbindet sich mit dem Resource Adapter und leitet die Parameterwerte an diesen weiter. Der Resource Adapter stellt das JCA-Interface für den JCA Connector zur Verfügung und ist mit der zu integrierenden Applikation verbunden.
3. Der Resource Adapter leitet die Parameter zur Verarbeitung an die Applikation weiter.
4. Wenn die Applikation bei der Bearbeitung der Parameter Rückgabewerte erzeugt, dann werden diese über den Resource Adapter an den JCA Connector übergeben. Der JCA Connector gibt die Rückgabewerte als XML-basierte Nachricht aus.

18.2 Eingangsnachricht erstellen

Zum Erstellen einer Eingangsnachricht verwenden Sie den JCA Connector Explorer. Beim Erstellen legen Sie eine InteractionSpec-Klasse und ein Record-Datensatzobjekt fest.

So gehen Sie vor

1. Erstellen Sie einen XSLT Converter.
2. Öffnen Sie im Bereich „XML-Zieldatei“ das ⬇ -Menü.
3. Wählen Sie „Öffnen von > JCA Explorer“. Der Assistent öffnet sich.
4. Wählen Sie eine der beiden Optionen:
   - **JCA Library auswählen** (*.jar)

Wählen Sie diese Option, wenn Sie den Namen der JCA Library kennen. Diese Auswahl schränkt die Klassen auf diejenigen ein, welche die Library anbietet, und erleichtert damit das Auffinden der passenden Einträge.

- **JCA InteractionSpec-Klasse auswählen**

  Wählen Sie diese Option, wenn Sie den Namen der JCA Library nicht kennen.

  Mit der InteractionSpec-Klasse werden die auf dem JCA Resource Adapter aufzurufenden Funktionen festgelegt.

5. Klicken Sie auf „Weiter" und lassen Sie sich von dem Assistenten durch die nächsten Dialoge führen.
6. **Record festlegen**

   Wählen Sie den Typ des Datensatzes aus, der als Input-Parameter dienen soll und benennen Sie diesen:

   - **MappedRecord**: Liste, deren Werte über einen alphanumerischen Index referenziert sind
   - **IndexedRecord**: Liste, deren Werte über einen numerischen Index referenziert sind
7. **Methode einfügen**
   a. Wählen Sie im Baum den Knoten „Methods" aus.
   b. Öffnen Sie das Kontextmenü und wählen Sie „Methode hinzufügen". Ein Dialog öffnet sich.
   c. Wählen Sie alle Methoden, die aufgerufen werden sollen.
   d. Klicken Sie auf „OK", um zurück in die Baumansicht zu gelangen.

   Die gewählten Methoden wurden hinzugefügt und alle Parameter unterhalb der Methodenknoten gesetzt.
8. **Parameterwerte setzen**

   Um einen vollständigen Methodenaufruf absetzen zu können, müssen Sie die Parameter mit konkreten Werten füllen.

   a. Markieren Sie einen Parameter.
   b. Öffnen Sie das Kontextmenü und wählen Sie „Wert setzen". Ein Dialog öffnet sich. In diesem Dialog wird der Wert für den gewählten Parameter gesetzt.

      Zur Unterstützung wird der Datentyp des Parameters angezeigt. Wenn es sich bei dem Parameter um ein Java-Objekt mit einem komplexen Datentyp handelt, dann muss zunächst ein passender Constructor zum Erzeugen dieses Java-Objektes ausgewählt werden.
9. Wenn alle Einstellungen vollständig sind, beenden Sie den Assistenten. Die Struktur der Eingangsnachricht wird erzeugt und angezeigt.

10. Ziehen Sie die Struktur ins XSLT-Template, um das XSLT-Stylesheet zu erstellen.
    → Für Hinweise über die Verwendung des XSLT Converters siehe *Technical Workflow erstellen 3: CSV-XML konvertieren nach openTRANS (Tutorials, Kap. 6.5, S. 116)*.

18.3 Beispiel-Eingangsnachricht

Der JCA Connector stellt die Variablen `cf` und `ix` bereit, die in der Eingangsnachricht verwendet werden können:

- `$cf` bietet Zugriff auf eine Instanz des Interfacetyps `javax.resource.cci.ConnectionFactory`.
- `$ix` steht für den Zugriff auf eine Instanz des Interfacetyps `javax.resource.cci.Interaction` bereit.

→ Für detaillierte Informationen über die Struktur der Ein- und Ausgangsnachrichten siehe *Ein- und Ausgangsnachrichten-Struktur (Workbench/Process Engine: Systemkonnektor-Guide, Kap. , S. 177)*.

Beispiel-Eingangsnachricht

Listing

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <jca>
 3   <!--References:-->
 4   <!--cf - javax.resource.cci.ConnectionFactory class from JCA Connector-
   ->
 5   <!--ix - javax.resource.cci.Interaction class to execute JCA call on-->
 6   <object name="iSpec">
 7     <com.dsoft.jca.eis.EisInteractionSpec>
 8       <a>a</a>
 9       <functionName>functionName</functionName>
10       <schema>schema</schema>
11     </com.dsoft.jca.eis.EisInteractionSpec>
12   </object>
13   <setA object="$iSpec" result="">
14     <param type="java.lang.String" name="">
15       <string>a</string>
16     </param>
```

/Listing (Abschnitt 1 von 2)

Listing

```
17   </setA>
18   <getRecordFactory object="$cf" result="rf"/>
19   <createIndexedRecord object="$rf" result="iRec">
20     <param type="java.lang.String">
21       <java.lang.String>InputRecord</java.lang.String>
22     </param>
23   </createIndexedRecord>
24   <execute object="$ix" result="rec">
25     <param type="javax.resource.cci.InteractionSpec" object="$iSpec"/>
26     <param type="javax.resource.cci.Record" object="$iRec"/>
27   </execute>
28   <result value-of="$rec"/>
29 </jca>
```

/Listing (Abschnitt 2 von 2)

Ausgeführter Java-Code

Mit der Beispiel-Eingangsnachricht wird folgender Java-Code ausgeführt:

Listing

```
 1 // following code is covered by JCA module settings
 2 ConnectionFactory cf = …;
 3 Connection con = cf.getConnection();
 4 Interaction ix = con.getInteraction();
 5 // following code is covered by JCA input message
 6 EisInteractionSpec iSpec = new EisInteractionSpec();
 7 iSpec.setA("a");
 8 RecordFactory rf = cf.getRecordFactory();
 9 IndexedRecord iRec = rf.createIndexedRecord("InputRecord");
10 Record rec = ix.execute(iSpec, iRec);
```

/Listing

Ausgangsnachricht

Die Ausgangsnachricht enthält den Wert der Variablen `rec`.

18.4 Dialog „J2EE Connector Architecture Adapter“

In diesem Dialog konfigurieren Sie die JCA Connector Eigenschaften.

Allgemeine Einstellungen

- **Connection Factory Lookup-Name (JNDI)**
  Name der JCA Adapter Connection Factory an, z. B. `java:/eis/PropertiesFileAdapter`.
  Mit der Connection Factory legen Sie fest, zu welchem Resource Adapter die Verbindung hergestellt werden soll.
  Die Connection Factory wird vom Resource Adapter zur Verfügung gestellt und über das JNDI dem JBoss bekannt gemacht.

Authentifizierung

- **Authentifizierung erforderlich**
  Um festzulegen, dass eine Authentifizierung gegenüber dem Resource Adapter erforderlich ist.
- **ConnectionSpec Klassenname**
  Klassenname der ConnectionSpec.
  Um die Authentifizierungsdaten an die Methode `getConnection` der ConnectionFactory zu übergeben.
  Wenn die Angabe fehlt, dann wird die Verbindung zum Resource Adapter ohne Parameter hergestellt und keine Authentifizierung vorgenommen.
- **Benutzername**: Benutzernamen für den Resource Adapter.
- **Passwort**: Passwort für den Resource Adapter.

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

Ausgabe-Optionen

Nur Resultate

Diese Option verbessert die Übersichtlichkeit und Performance bei großen XML-Eingabenachrichten. Wenn markiert, dann enthält die Ausgabenachricht nur das XML-Element `<result>`. Die Eingabenachricht wird nicht ausgegeben.

19 JMS Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Der JMS (Java Message Service) Connector verbindet die inubit Process Engine mit einem JMS Provider.

Konnektortypen

Abhängig von seiner Konfiguration bietet der Konnektor folgende Funktionen:

- **Input Connector**
  Holt zeitgesteuert Nachrichten von einem JMS Provider und übergibt sie an das nachfolgende Modul im Technical Workflow.
- **Input Listener Connector**
  Wartet auf Nachrichten von einem JMS Provider. Sobald diese eingetroffen sind, startet der Connector den Technical Workflow, indem er sie an das nachfolgende Modul übergibt.
- **Medium Connector**
  Wird durch den Empfang einer Nachricht vom vorhergehenden Modul im Workflow gestartet, holt Nachrichten von einem JMS Provider ab und übergibt diese an das nachfolgende Modul im Workflow.
- **Medium Listener Connector**
  Wird durch den Empfang einer Nachricht vom vorhergehenden Modul im Workflow gestartet und wartet dann auf Nachrichten von einem JMS Provider. Sobald eine Nachricht eintrifft, wird diese an das nachfolgende Modul übergeben.
- **Output Connector**
  Sendet Nachrichten aus dem Technical Workflow an einen JMS Provider.

→ Siehe auch

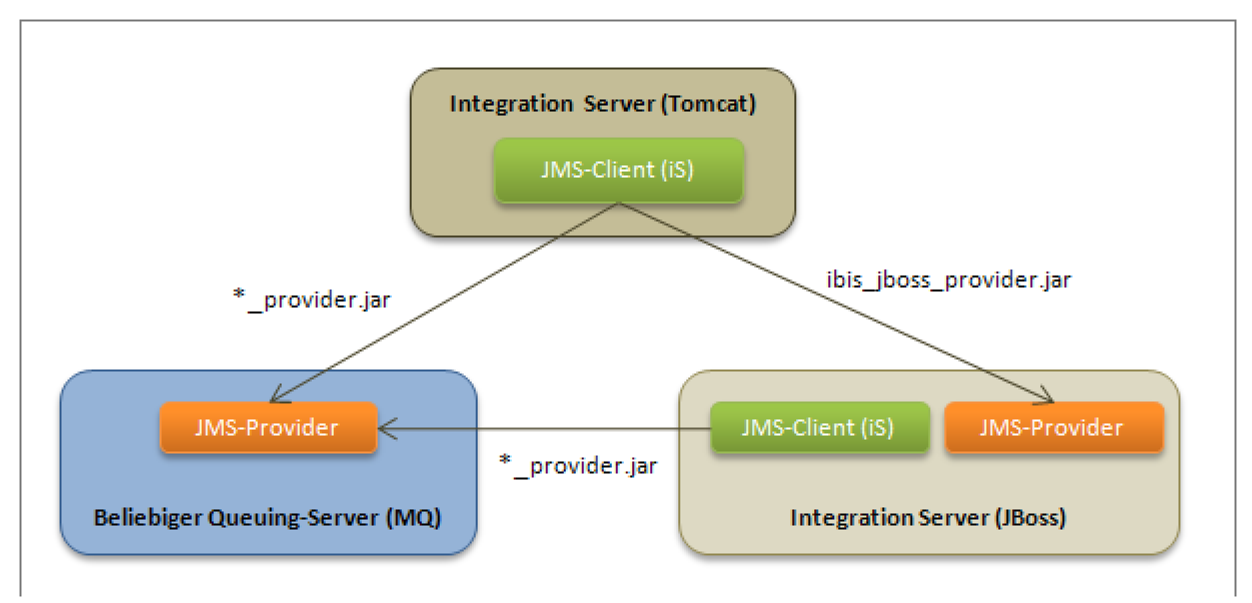
- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

19.1 Zugriff auf JMS-basierten Queuing-Server konfigurieren

Übersicht

Für den Einsatz des JMS Connectors benötigen Sie einen JMS-basierten Applikationsserver mit Queuing-Schnittstelle wie z. B. den integrierten JBoss der inubit Suite 6 Enterprise Edition.

Wie Sie den Zugriff auf diesen JMS-basierten Applikationsserver konfigurieren, hängt davon ab, welche Edition der inubit Suite 6 Sie nutzen:



- **inubit Process Engine mit Tomcat (Entry, Standard oder Professional Edition)**
  Sie können den JBoss-Applikationsserver einer entfernten Enterprise oder Enterprise Plus Edition oder einen beliebigen anderen JMS-basierten Queuing-Server nutzen.
  In beiden Fällen benötigen Sie eine zusätzliche jar-Datei als Kommunikationsbrücke.
  → Siehe *JMS Connector in Entry, Standard oder Professional Edition einsetzen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 19.1.1, S. 189).*
- **inubit Process Engine mit JBoss (Enterprise oder Enterprise Plus Edition)**

Sie können den JMS Connector ohne weitere Konfigurationen einsetzen, weil der JMS Connector die Queuing-Schnittstelle des integrierten JBoss-Applikationsservers oder den entfernten JBoss einer anderen Enterprise oder Enterprise Plus Edition nutzen kann.

Alternativ können Sie einen beliebigen, eigenen JMS-basierten Queuing-Server einsetzen.

→ Siehe *JMS Connector in Enterprise/Enterprise Plus Edition mit beliebigem JMS-basierten Queuing Server einsetzen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 19.1.2, S. 190).*

19.1.1 JMS Connector in Entry, Standard oder Professional Edition einsetzen

Wenn Sie eine Entry, Standard oder Professional Edition (basierend auf dem Tomcat-Applikationsserver) einsetzen, können Sie entweder die Queuing-Schnittstelle des JBoss-Applikationsserver in einer entfernten Enterprise oder Enterprise Plus Edition nutzen oder einen anderen, beliebigen JMS-basierten Queuing-Server. Dazu benötigen Sie zusätzlich eine jar-Datei als Kommunikationsbrücke zwischen Ihrer inubit Process Engine und dem entfernten Queuing-Server.

So gehen Sie vor

1. `*.jar`-Datei anfordern:
   - Wenn Sie auf den JBoss einer Enterprise oder Enterprise Plus Edition zugreifen möchten, fordern Sie die Datei `ibis_jboss_provider.jar` beim Support der inubit AG an (support@inubit.com).
   - Wenn Sie auf einen beliebigen JMS-basierten Queuing-Servers zugreifen möchten, fordern Sie die Datei `*-provider.jar` von Ihrem JMS-Provider an.
2. Kopieren Sie die Datei in das Verzeichnis `<iS-installdir>/Tomcat/webapps/ibis/WEB-INF/lib`.
3. Starten Sie Tomcat neu.
4. Geben Sie beim Konfigurieren des JMS Connectors im *Dialog „Verbindungskonfiguration“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 19.2.1, S. 191)* im Feld „URL des JNDI-Servers“ die URL des entfernten Queuing-Servers ein.

19.1.2 JMS Connector in Enterprise/Enterprise Plus Edition mit beliebigem JMS-basierten Queuing Server einsetzen

Um den integrierten JBoss-Applikationsserver Ihrer Enterprise oder Enterprise Plus Edition oder einen entfernten JBoss einer anderen Enterprise oder Enterprise Plus Edition als Queuing-Schnittstelle zu nutzen, sind keine weiteren Schritte nötig.

Wenn Sie die Queuing-Schnittstelle eines beliebigen anderen JMS-basierten Queuing-Servers nutzen möchten, benötigen Sie zusätzlich eine jar-Datei als Kommunikationsbrücke zwischen Ihrer inubit Process Engine und dem entfernten Queuing-Server.

So gehen Sie vor

1. Fordern Sie die Datei `*-provider.jar` bei Ihrem JMS-Provider an.
2. Kopieren Sie die Datei in das Verzeichnis `<iS-installdir>server/JBoss/server/default/deploy/ibis.ear/lib`.
3. Starten Sie JBoss neu.
4. Geben Sie beim Konfigurieren des JMS Connectors im *Dialog „Verbindungskonfiguration“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 19.2.1, S. 191)* im Feld „URL des JNDI-Servers“ die URL des entfernten Queuing-Servers ein.

19.2 Dialogbeschreibungen

Dieser Abschnitt erläutert die folgenden Themen:

19.2.1 Dialog „Verbindungskonfiguration“

In diesem Dialog geben Sie die Eigenschaften an, die für Verbindungen zu einem JNDI (Java Naming and Directory Interface) Provider gebraucht werden. JMS benutzt das JNDI, um die benötigten Ressourcen zu finden.

Grundkonfiguration

- **Voreingestellte JNDI-Konfiguration benutzen**
  Zu markieren, wenn Sie die Datei `jndi.properties` in Ihrem Classpath haben und die dort angegebenen Eigenschaften verwenden möchten.
- **URL/Prefix/Naming Context Factory**
  Standardwerte für die gängigsten JMS Provider:
  - **JBoss**
    URL = `localhost:1099`
    Prefix = `org.jboss.naming`
    Naming Context Factory = `org.jnp.interfaces.NamingContextFactory`
  - **MQSeries**
    URL = `ldap://<name>:389/<LDAP DN>`
    Prefix = `com.ibm.jndi`
    NamingContextFactory = `com.sun.jndi.ldap.LdapCtxFactory`
  - **SonicMQ**
    ContextProviderUrl = `file://localhost/d:sonicmq`
    ContextUrlPrefixes = `<empty>`
    NamingContextFactory=`com.sun.jndi.fscontext.RefFSContextFactory`
  - **WebLogic**
    ContextProviderUrl = `t3://localhost :7001`
    ContextUrlPrefixes = `<empty>`
    NamingContextFactory = `weblogic.jndi.WLInitialContextFactory`
- **Weitere JNDI Eigenschaften**
  Geben Sie beliebige JNDI-Properties als Property-Wert-Paare an:

```
com.sonicsw.jndi.mfcontext.domain=Domain2
com.sonicsw.jndi.mfcontext.idleTimeout=6000
```

  Mehrere Properties trennen Sie mit jeweils einem Leerzeichen.

Bei Problemen wenden Sie sich an den Systemadministrator Ihres JMS Providers!

Mitteilungstyp

- Nachrichten können in folgenden Formaten versendet/empfangen werden:
  - **TextMessage**
    Der Payload sind Daten, die als Zeichenkette gespeichert wurden. Zum Austausch von einfachen Textnachrichten und komplexer Zeichendaten wie XML-Dokumente.
  - **ObjectMessage**
    Überträgt als Payload ein serialisierbares Java Object. Zum Austausch von Java-Objekten.
  - **BytesMessage**
    Der Payload wird als Byte-Array gespeichert. Nützlich für den Austausch von Daten im nativen Format einer Applikation und für den Fall, dass JMS als Transportmittel zwischen zwei Systemen verwendet wird, bei denen der JMS Client den Payload-Typ der Nachrichten nicht kennt.
  - **StreamMessage**
    Sequenz primitiver Java-Typen. Das Nachrichtenobjekt überwacht deren Anordnung und Typen im Strom. Datenkonvertierungsbeschränkungen sind gültig, daher ist es z. B. ein Fehler, wenn eine JMS-Applikation versucht, einen Double-Wert als short-Wert zu lesen.

    Siehe auch die JMS Spezifikation, Version 1.1, Abschnitt „Message Selector“, unter http://java.sun.com/products/jms/docs.html

    Beispiel: 21ABCDEFGH32.345

    Besteht aus folgenden drei Feldern:
    - Integer: 21
    - String: ABCDEFGH
    - Float: 32.345

    Wenn die Datenstruktur unbekannt ist, dann kann die generische Methode `readObject()` verwendet werden, um das nächste Objekt im Strom zurückzugeben.
    Wenn die Datenstruktur bekannt ist, dann kann der JMS Client den Objekttyp, auf den zugegriffen werden soll, angeben.
  - **MapMessage**
    Der Payload einer MapMessage wird als eine Menge von Name/Werte-Paaren gespeichert. Der Name ist eine Zeichenkette und der Wert ist typisiert. Wird verwendet um Schlüssel-Werte zu übertragen die sich von einer Nachricht zur anderen ändern können.

    Beispiel: NumberOfCopies:5

    Dabei ist NumberOfCopies der Schlüssel und 5 der Wert.

Die Daten werden über die Methode `getMapNames()` aufgerufen, welche ein Java Enumeration-Objekt zurück gibt. Mit der Methode `hasMoreElements()` können Sie über eine MapMessage iterieren.

- **Zeichensatzkodierung**
  Wählen Sie eine Kodierung aus der Liste aus.

Authentifizierung

- **Anonymer Login**
  Zu markieren, wenn der JMS Provider keine Authentifizierung vom Client erwartet.
- **Login/Passwort**
  Anmeldedaten des Clients am JMS Provider.

Sicherheit

- **Sicherheitsmanager benutzen**
  Zum Aktivieren des Security Managers.
  Security Manager sind abhängig vom eingesetzten JMS/JNDI Provider und sollten nur verwendet werden, wenn sie laut den Vorgaben Ihres JMS Providers nötig sind.
- **Security Manager**
  Name der Klasse, in welcher der Security Manager implementiert ist. Diese Klasse ist in den `*.jar`-Dateien Ihres JMS-Clients enthalten.

19.2.2 Dialog „Kommunikationsmodell“

In diesem Dialog haben Sie folgende Optionen:

Kommunikationsmodell

Wählen Sie eines der Modelle:

- **Publish/Subscribe**
  Dieses Modell wird für allgemeine Broadcast-Applikationen verwendet, bei denen eine Nachricht viele Empfänger hat:
  Ein Nachrichten-Lieferant versendet Nachrichten zu einem Topic an alle Empfänger, die sich vorher an dem Topic eingeschrieben haben.
  Empfänger, die zum Zeitpunkt des Versendens nicht eingeschrieben waren, erhalten keine Nachrichten.

Sonderfall „Dauerhafte Einschreibung“:
Die Empfänger müssen sich einschreiben, aber nicht ständig aktiv

auf Nachrichten warten, sondern können sich diese später abholen.

- **Point-to-Point**

  Dieses Modell wird verwendet, wenn Nachrichten nur einen Empfänger haben:

  Die Nachrichten, die ein Nachrichten-Lieferant schickt, werden first in/first out in eine Nachrichten-Queue gestellt.

  Nachrichten-Empfänger können die Nachrichten der Reihe nach zu einem beliebigen, späteren Zeitpunkt aus der Queue abholen.

Einstellungen

Die folgenden Optionen werden angezeigt, wenn Sie als Kommunikationsmodell „Point-to-Point“ gewählt haben.

- **Name der Queue**

  Der Queue Name bezeichnet ein Objekt, das vom JMS Provider erzeugt und beim JNDI-Server registriert wurde. Der JMS Connector durchsucht den JNDI-Server nach dem angegebenen Queue Name und erhält eine Referenz auf das Objekt.

- **Klasse für Verbindungsaufbau**

  Name der Connection Factory. Die Connection Factory ist das Objekt, das ein Client benutzt, um eine Verbindung mit dem JMS Provider zu erstellen.

  Das Objekt wurde von dem JMS Provider erzeugt und beim JNDI-Server registriert. Der JMS Connector durchsucht den JNDI-Server nach der angegebenen Klasse und erhält eine Referenz auf das registrierte Objekt.

- **Synchroner Modus**

  Wenn markiert, dann werden Nachrichten vom JMS-Server synchron empfangen, d.h. der Eingang einer Nachricht startet den Workflow, der bis zum Ende durchlaufen wird, bevor die nächste Nachricht empfangen werden kann. Die laufende Workflow-Verarbeitung blockt den Empfang neuer Nachrichten und erst der beendete Workflow gibt das Signal zum Empfang und der Verarbeitung der nächsten Nachricht in der Queue. Falls nicht markiert, läuft der Nachrichten-Empfang asynchron ab und Nachrichten werden an den Workflow übergeben, sobald sie eingetroffen sind.

Die folgenden Optionen werden angezeigt, wenn Sie als Kommunikationsmodell „Publish/Subscribe“ gewählt haben.

- **Topic-Name**

  Ein Topic ist eine Zeichenkette, welche die Art der Daten beschreibt, die in einem Publish/Subscribe-System publiziert sind.

Der Topic-Name bezeichnet ein Objekt, das vom JMS Provider erzeugt und beim JNDI-Server registriert wurde. Der JMS Connector durchsucht den JNDI-Server nach dem angegebenen Topic-Name und erhält eine Referenz auf das Objekt.

- **Klasse für den Verbindungsaufbau**

  Name einer Connection Factory. Diese ConnectionFactory ist ein Objekt, das ein Client benutzt, um eine Verbindung mit dem JMS Provider zu erstellen.

  Das Objekt wird vom JMS Provider erzeugt beim JNDI-Server registriert. Der JMS Connector durchsucht den JNDI-Server nach dem angegebenen Namen und erhält eine Referenz auf das registrierte Objekt.

- **Topic No Local**

  Blockiert den Empfang von Nachrichten, die über die eigene Verbindung ausgeliefert wurden.

- **Message Acknowledgment**

  Zur Auswahl des Kontroll-Levels bei der Bestätigung von Nachrichten. Es gibt folgenden Bestätigungsarten:

  - AUTO_ACKNOWLEDGE: Die JMS-Sitzung bestätigt automatisch, dass der Client die Nachricht erhalten hat. Entweder wenn der Client erfolgreich einen Aufruf zum Nachrichtenerhalt durchgeführt hat oder wenn der Aufruf des Listeners zum Empfang einer Nachrichten geführt hat.
  - CLIENT_ACKNOWLEDGE: Bestätigung auf Session-Ebene. Ein Client bestätigt eine Nachricht durch den Aufruf der Bestätigungsmethode der Nachricht.
  - DUPS_OK_ACKNOWLEDGE: Weist die JMS Session an, die Auslieferung einer Nachricht „lazy“ zu bestätigen. Das führt im Allgemeinen zur Auslieferung einer weiteren Kopie der Nachricht, wenn der JMS Provider nicht direkt bestätigt.

    Sollte nur von Clients verwendet werden, die Nachrichtenkopien zulassen.

Dauerhafte Subskription

(Nur, wenn das Kommunikationsmodell „Publish/Subscribe“ gewählt ist)

Die dauerhafte Subskription ist ein Sonderfall des Publish-Subscribe-Kommunikationsmodells. Empfänger müssen sich beim JMS Provider einschreiben, aber nicht ständig auf Nachrichten warten. Stattdessen können sie diese später abholen.

- **Subskriptions-ID**: ID des Clients für die Verbindung
- **Login**: Nutzername des Clients
- **Passwort**: Passwort

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

19.2.3 Dialog „Nachrichten-Selektor“

In diesem Dialog haben Sie folgende Optionen:

Nachrichten-Selektor

Nachrichten-Selektor werden genutzt, um gezielt Nachrichten aus einem Topic oder einer Queue auszuwählen. Wenn ein Nachrichten-Selektor angegeben ist, dann werden nur Nachrichten an den Client übergeben, deren Header und Properties zu dem Nachrichten-Selektor passen.

Nachrichten-Selektoren können nicht auf Werte im Nachrichten-Body zugreifen!

Ein Nachrichten-Selektor ist ein bedingter Ausdruck bestehend aus einem Identifier und einem Operator gefolgt von einem Literal. Die folgende Tabelle zeigt, wie Nachrichten-Selektoren konstruiert werden:

Element	Gültige Werte
Identifier	Property oder Header-Feldreferenz
Operatoren	AND, OR, LIKE, BETWEEN, =, <>, <, >, <=, >=, IS NULL, IS NOT NULL
Literale	▪ boolean ▪ byte ▪ short ▪ int ▪ long ▪ float ▪ double ▪ String

Siehe auch die JMS Spezifikation, Version 1.1, Abschnitt „Message Selector“, unter http://java.sun.com/products/jms/docs.html

19.2.4 Dialog „Zusätzliche Eigenschaften“

In diesem Dialog können Sie Eigenschaften selbst definieren. Diese Eigenschaften können Sie nutzen, um die Kompatibilität zu anderen Nachrichtensystemen herzustellen oder Nachrichten-Selektoren zu definieren.

Um eine Eigenschaft hinzuzufügen, klicken Sie auf den Button „Eigenschaften hinzufügen“. Füllen Sie im angezeigten Dialog die Eingabefelder „Name“, „Typ“ und „Wert“.

→ Siehe *Dialog „Nachrichten-Selektor“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 19.2.3, S. 196).*

20 LDAP Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Mit dem LDAP Connector können Sie Daten auf einem Lightweight Directory Access Protocol (LDAP)-Server einfügen, ändern, löschen und suchen.

Konnektortypen

Ein LDAP Connector kann als Medium- oder Outputkonnektor konfiguriert werden:

- **Medium Connector**
  Als Mediumkonnektor erhält der LDAP Connector vom Vorgänger eine Abfrage als DSML-Nachricht, sendet diese an den LDAP-Server, erhält vom LDAP-Server ein Ergebnis und gibt dieses Ergebnis an das nachfolgende Modul weiter.
- **Output Connector**
  Als Outputkonnektor erhält der LDAP Connector vom Vorgänger-Modul eine DSML-Nachricht und sendet diese an den LDAP-Server.

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

20.1 Funktionsprinzip

Der LDAP Connector stellt die Verbindung zu einem LDAP-Server her, liest einen DSMLv2 Request und übersetzt die darin enthaltenen Anweisungen in LDAP-Aufrufe.

Die Directory Services Markup Language (DSML) ist ein OASIS-Standard, der den Zugriff auf Verzeichnisse mittels XML-Schema und SOAP als Transportmechanismus spezifiziert und das komplette LDAP-Datenmodell abbildet.

Um die Verbindung zu einem LDAP-Server aufzubauen und Abfragen auszuführen, müssen Sie

1. einen LDAP Connector mit Verbindungsinformationen zum LDAP-Server erstellen
2. mit Hilfe eines XSLT Converters einen DSMLv2-Request in folgender Syntax erstellen:

```
<batchRequest
 xmlns="urn:oasis:names:tc:DSML:2:0:core">
 <-- Jede einzelne Anweisung muss ein
     DN-Attribut enthalten! -->
 <modifyRequest
   dn="CN=Joe Smith, OU=Dev, DC=inubit,
   DC=com>…
 </modifyRequest>
 <addRequest>…</addRequest>
 <delRequest>…</delRequest>
</batchRequest>
```

3. Die Response ist abhängig von der Konfiguration des Connectors:
   - Bei einem Output Connector werden Fehler- bzw. Erfolgsmeldungen zurückgegeben.
   - Bei einem Medium Connector werden die Ergebnisse in Form einer DSMLv2 Response zurückgesendet:

```
<batchResponse
xmlns="urn:oasis:names:tc:DSML:2:0:core">
<modifyResponse>…</modifyResponse>
<addResponse>…</addResponse>
<delResponse>…</delResponse>
</batchResponse>
```

20.2 DSMLv2-Anweisungen in Requests

Dieser Abschnitt erläutert folgende DSMLv2-Anweisungen:

- *Modify, S. 201*
- *Search, S. 202*
- *Add, S. 203*
- *Delete, S. 203*

Für alle Zugriffe auf konkrete Einträge eines LDAP-Servers benötigen Sie die eindeutigen Namen (distinguished names) der entsprechenden Objekte!

DN-Attribute in Anweisungen

Jede Anweisung in einem Request muss ein DN-Attribut (distinguished names) enthalten. Das DN-Attribute wird benötigt, um einen Eintrag in einem LDAP-Verzeichnis eindeutig zu identifizieren und beschreibt, wo genau in der Verzeichnishierarchie sich der Eintrag befindet.

Siehe auch

- DSML-Dokumentation unter http://www.oasis-open.org/committees/dsml/docs/DSMLv2.doc
- DSML-Schema unter http://www.oasis-open.org/committees/dsml/docs/DSMLv2.xsd
- vollständige Definition der Distinguished Names unter http://www.ietf.org/rfc/rfc2253.txt
- LDAP Schema Viewer unter http://ldap.akbkhome.com/index.php, der eine praktische Schnittstelle für die Untersuchung von Standard-LDAP-Schema-Objekten ist.

20.2.1 Modify

In DSMLv2 werden alle Änderungen an Attributen durch Anhängen eines `operation` Attributs an ein `attr` Element spezifiziert. Eine `operation` kann `add`, `delete` oder `replace` sein.

Beispiel

Die folgende Anweisung aktualisiert die Telefonnummer des Mitarbeiters Bob Rush:

```
<modifyRequest
  dn="CN=Bob Rush,OU=Dev,DC=Example,DC=COM">
  <modification
    name="telephoneNumber"
    operation="replace">
    <value>536 354 2343</value>
    <value>234 212 4534</value>
  </modification>
</modifyRequest>
```

20.2.2 Search

Die Anweisung `searchRequest` sucht nach Daten auf dem LDAP-Server.

Beispiel „Suchen und Ausgeben“

Es werden alle Personen mit dem Namen „John“ gesucht, die sich im Pfad „ou=Marketing, dc=inubit, dc=com“ befinden. Dabei werden alle gefunden Objekte samt ihrer Attribute ausgegeben.

```
<searchRequest
  dn="ou=Marketing,dc=inubit,dc=com"
  scope="singleLevel"
  derefAliases="neverDerefAliases"
  sizeLimit="1000">
  <filter>
    <equalityMatch name="cn">
      <value>john</value>
    </equalityMatch>
  </filter>
```

Beispiel „Suchen und Ausgabe eingrenzen“

Im folgenden Beispiel wird nach den gleichen Kriterien wie oben gesucht. Allerdings sollen die ausgegeben Attribute auf das Attribut „objectSid“ beschränkt werden.

```
<searchRequest
  dn="ou=Marketing,dc=inubit,dc=com"
  scope="singleLevel"
  derefAliases="neverDerefAliases"
  sizeLimit="1000">
  <filter>
    <equalityMatch name="cn">
      <value>john</value>
    </equalityMatch>
  </filter>
  <attributes>
    <attribute name="objectSid" type="binary"/>
  </attributes>
```

Wenn binäre Attribute über den LDAP-Connector ausgelesen werden sollen, müssen diese Attribute bereits in der Eingangsnachricht gekennzeichnet bzw. maskiert werden. Im Search-Request müssen Sie dazu das Attribut `type='binary'` einfügen. Die Werte aus dem LDAP-System für binäre Attribute werden mit der base64-Kodierung in der XML-Ausgangsnachricht abgelegt.

20.2.3 Add

Neue Objekte und Attribute fügen Sie mit `<addRequest>` ein.

Beispiel

Die Anweisung zum Einfügen eines Objektes „Alice Johnson“ vom Typ „Person“ lautet:

```
<addRequest
    dn="OU=Marketing,DC=inubit,DC=com">
  <attr name="objectclass">
    <value>person</value>
  </attr>
  <attr name="objectclass">
    <value>organizationalPerson</value>
  </attr>
  <attr name="sn">
    <value>Johnson</value>
  </attr>
  <attr name="givenName">
    <value>Alice</value>
  </attr>
  <attr name="title">
    <value>Software Design Engineer</value>
  </attr>
</addRequest>
```

20.2.4 Delete

Zum Löschen von Daten auf dem LDAP-Server wird die Anweisung `<delRequest>` verwendet.

Beispiel

Die Anweisung löscht das Objekt „Alice“ aus der LDAP-Datenbank:

```
<delRequest dn="cn=Alice, ou=Marketing,
dc=inubit,dc=com"/>
```

20.3 Dialog „LDAP Connector Eigenschaften“

In diesem Dialog haben Sie folgende Optionen:

LDAP Server

URL

Ersetzen Sie `<hostname>` durch den Namen des LDAP-Servers.

Wenn der Verbindungstest mit dieser Portnummer fehlschlägt, erfragen Sie die Portnummer des LDAP-Servers bei dem Administrator des LDAP-Servers.

Authentifizierung

- **Anonymes Login**
  Wählen Sie diese Option, wenn der LDAP-Server ein anonymes Login unterstützt.
- **Login**
  Wenn Sie kein anonymes Login verwenden, geben Sie Ihre Benutzerkennung für den LDAP-Server an.
- **Passwort**
  Geben Sie das zu Ihrer Benutzerkennung gehörende Passwort an.

Namensdienst

Klassenname

Vorbelegt mit der Standard-Java Klasse, welche mit der inubit Suite 6 ausgeliefert wird. Um eine andere Klasse für den LDAP-Namens- und Verzeichnisdienst zu verwenden, geben Sie den Klassennamen ein.

Stellen Sie sicher, dass die angegebene Klasse im Java-Classpath vorhanden ist!

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

21 Livelink PDMS Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Mit dem Livelink PDMS Connector verbinden Sie die inubit Process Engine mit einem Livelink Archive Server. Sie können mit dem Konnektor auf archivierte Daten und Dokumente im „Production Document Management System“ (PDMS) zugreifen bzw. Daten dort ablegen.

Konnektortypen

Ein Livelink Connector wird immer als Mediumkonnektor eingesetzt.

→ Siehe auch
- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

21.1 JAR-Dateien installieren

Sie müssen folgende externe `*.jar`-Dateien auf der inubit Process Engine im Verzeichnis `<iS-installdir>/server/lib/ext` installieren. Die Dateien erhalten Sie vom Hersteller des PDMS:

- `dmsclient.jar`
- `iaik_javax_crypto.jar`
- `iaik_jce_full.jar`
- `iaik_jsse.jar`
- `iaik_ssl.jar`
- `ixosBase.jar`
- `ixosCat.jar`
- `jhttp.jar`
- `jiaik.jar`

- `jsec.jar`
- `ot-commons-httpclient.jar`
- `spheon-jsoap.jar`
- `w3c_http.jar`

→ Siehe *Treiber installieren (Process Engine: Administrator- und Entwickler-Guide, Kap. 3.4, S. 51).*

21.2 Inhalt und Struktur der Eingangs- und Ausgangsnachrichten

Dieser Abschnitt erläutert die folgenden Themen:

Als Eingangsnachricht benötigt der Livelink Connector eine XML-Datei mit folgendem Inhalt:

- **Kommandos** zur Steuerung der Aktionen am Livelink Archive Server
- **Entitäten** als Vorgaben für anzulegende Objekte oder als Vorgabe, welche Eigenschaften an bestehenden Entitäten geändert werden sollen.
- **Bedingungen** für die Suche nach spezifischen Eigenschaften, Dokumentklassen oder Dokumenten-IDs.

Die Ausgangsnachricht hat dieselbe Struktur wie die Eingangsnachricht.

Struktur der Eingangsnachricht

```
<LivelinkArchivingConnector>
  <Commands>
    <Command Type="<!-- Kommando -->"
      DetailedResult="true|false">
      <Entities>
        <Entity/> <!-- Mind. eine Entität,
        je nach Kommando -->
      </Entities>
      <Condition/>
    </Command>
  </Commands>
</LivelinkArchivingConnector>
```

Struktur der Ausgangsnachricht

Die Ausgangsnachricht liefert die durch Kommandos adressierten Entitäten zurück.

```
<LivelinkArchivingConnector>
  <Entities> <!-- Entitäten des Kommandos 1 -->
    <Entity/> <!-- 0 bis n Entitäten -->
  </Entities>
  <Entities> <!-- Entitäten des Kommandos 2 usw. -->
    <Entity/> <!-- 0 bis n Entitäten -->
  </Entities>
    <!-- Weitere Entitäten -->
</LivelinkArchivingConnector>
```

21.2.1 Kommandos

Es können pro Ausführung des Konnektors mehrere Kommandos übergeben werden. Diese werden in der definierten Reihenfolge und als eine Transaktion abgearbeitet.

Kommandotypen

Die folgende Tabelle gibt einen Überblick über die einzelnen Kommandotypen:

Kommando	Beschreibung
„Create“	Legt neue Entitäten mit allen übergebenen Eigenschaften auf dem Livelink Archive Server an.
"CreateIfNotExists"	Wie "Create", aber nur dann, wenn Entitäten noch nicht existieren. Ob Entitäten schon vorhanden sind, wird mittels der Bedingung auf dem Livelink Archive Server überprüft.

21.2.2 Bedingungen

Abhängig vom Kommando muss die Eingangsnachricht eine Such-Bedingung enthalten. Diese Bedingung wird in Form einer `where`-Klausel aufgestellt. Dabei wird für die `where`-Klausel die Livelink PDMS Query Language verwendet.

Einen Einstieg in die Livelink Query Language (LGQL) finden Sie unter https://ecommunities.belgium.be/ecomfrsupport/help/_en_US/websbroker/sr_lql.html.

Bedingung für spezifische Eigenschaft

```
<and>
  <equal>
    <property name="ixos.dms:Id"/>
    <literal value="aaps44v2vy6uqvorzmcje3enjexgo"
datatype="String"/>
  </equal>
</and>
```

Bedingung für spezifische Eigenschaft und Dokumentenklasse

```
<and>
  <equal casesensitive="true">
    <property name="ixos.bai:LinkId"/>
    <literal value="$$URN$$" datatype="String"/>
  </equal>
  <isa
entitytype="siemens.ad.bereichsarchiv:Dkl0007R3Lief
erung"/>
</and>
```

Bedingung für Dokument-ID

```
<and>
<equal>
<property name="ixos.dms:Id"/>
<literal value="aaps44v2vy6uqvorzmcje3enjexgo"
datatype="String"/>
</equal>
</and>
```

21.2.3 Entitäten

Dieser Abschnitt erläutert die folgenden Themen:

Die Entitäten sind die zentralen Elemente von Eingangs- und Ausgangsnachrichten und sind jeweils analog im Livelink Archive Server repräsentiert.

In der Eingangsnachricht dienen Sie abhängig vom Kommando als Vorgabe für anzulegende Entitäten oder dafür, welche Eigenschaften an bestehenden Entitäten geändert werden sollen.

In der Ausgangsnachricht dienen sie zur Übergabe der Eigenschaften.

Struktur der Entitäten

```
<Entities>
  <Entity>
    <!-- Entity-Eigenschaften -->
    [<Properties/>]
    [<Components/>]
  </Entity>
</Entities>
```

Inhalt in eckigen Klammern `[ ]` ist optional.

21.2.3.1 Entity-Eigenschaften

Entity-Eigenschaften sind primäre Attribute der Entität, z. B.:

```
<Entities>
  <Entity>
    <Type>siemens.ad.bereichsarchiv:Document</Type>
    [<Id>aabqfd4m4qsequf2heae3qmnjexgo</Id>]1)
    [<Version></Version>]1)
    <Properties/>
    <Notes/>
    <Annotations/>
    <Components/>
  </Entity>
</Entities>
```

Inhalt in eckigen Klammern `[ ]` ist optional.

Die Id (1) ist momentan nur als Rückgabewert implementiert.

Entity-Eigenschaft	Feld	Beschreibung
Type	Bedingt	Die Dokumentenklasse des anzulegenden Entities. Der Typ ist nur beim Neuanlegen (Kommando "Create*") relevant. Bei "Update" wird der Typ nicht beachtet.
ID	Optional	Legt neue Entitäten mit allen übergebenen Eigenschaften auf dem Livelink Archive Server an.
Version	Optional	Wie "Create", aber nur dann, wenn Entitäten noch nicht existieren. Ob Entitäten schon vorhanden sind, wird mittels der Bedingung auf dem Livelink Archive Server überprüft.

21.2.3.2 Entity-Properties

Entity-Properties beschreiben die Entität näher. Die Properties müssen im Livelink Archive Server für die entsprechende Dokumentenklasse (<Type>) definiert sein. Es müssen nur die Properties aufgeführt werden, die übergeben werden sollen.

```
<Entities>
  <Entity>
    <Properties>
      <Property
        Name="siemens.ad.bereichsarchiv:fk_systyp">
        11
      </Property>Property Name="ixos.bai:LinkedIds"
      MultiValue="true">
        <Value>gss:\\id1</Value>
        <Value>gss:\\id2</Value>
        <Value>myLink:\\foo\bar</Value>
      </Property>
    </Properties>
    <Notes/>
    <Annotations/>
    <Components/>
  </Entity>
</Entities>
```

Entity-Properties	Feld	Beschreibung
Property/@Name	Erforderlich	Der vollständige Name des Properties.
Property/@MultiValue	Optional	Kennzeichen, dass das Property aus mehreren Werten be-steht.
Property/Value	Bedingt	Ein oder mehrere Wert/e eines MultiValue-Properties.
Property	Erforderlich	Der Wert des Properties.

21.2.3.3 Entity-Components

Komponenten beinhalten die eigentlichen Dokumente (z. B. PDF-Dokumente). Pro Entität können mehrere Komponenten (also Dokumente) abgelegt werden. In der Regel ist dies aber immer dasselbe Dokument in unterschiedlichen Formaten. So kann z. B. eine Rechnung als TIFF-, als PDF- und als Word-Dokument angelegt werden. Die entsprechende Entität hätte dann 3 Komponenten.

```
<Entities>
```

```
    <Entity>
      <Properties/>
      <Notes/>
      <Annotations/>
      <Components>
        <Component>
          <Name>data</Name>
          <Format>application/pdf</Format>
          <Encoding>
          <CreatedBy></CreatedBy>
          <CreatedAt></CreatedAt>
          <LastModifiedBy></LastModifiedBy>
          <LastModifiedAt></LastModifiedAt>
          <Volume></Volume>
          <OnlineStatus></OnlineStatus>
          <Data>
          <!-- Base64 kodierte Binärdaten -->
          </Data>
        </Component>
      </Components>
    </Entity>
  </Entities>
```

Entity-Components	Feld	Beschreibung
Name	Erforderlich	Der Name der Komponente. Dieser Name muss pro Format eindeutig sein. Soll ein bestehendes Dokument überschrieben werden, ist der Name und das Format das entscheidende Kriterium (Primärschlüssel).
Format	Erforderlich	Der MIME-Type der Komponente. Soll ein bestehendes Dokument überschrieben werden, ist der Name und das Format das entscheidende Kriterium (Primärschlüssel).
Encoding	Optional	Das Encoding der Komponente.
CreatedBy	Bedingt	Der Wert des Properties.
CreatedAt	Bedingt	Nur lesend für Ausgangsnachricht.
LastModifiedBy	Bedingt	Nur lesend für Ausgangsnachricht.
LastModifiedAt	Bedingt	Nur lesend für Ausgangsnachricht.
Volume	Bedingt	Nur lesend für Ausgangsnachricht.
OnlineStatus	Bedingt	Nur lesend für Ausgangsnachricht.
Data	Erforderlich	Enthält das eigentliche Dokument mit base64-Kodierung. Im Archiv ist die Nachricht binär gespeichert. Sie wird vor und nach dem Speichern bzw. Lesen mit base64 kodiert.

21.2.4 Beispiel für Eingangsnachricht


```
<?xml version="1.0" encoding="utf-8"?>
<LivelinkArchivingConnector>
  <Commands>
    <Command Type="CreateIfNotExists" [DetailedResult="true|false"]>
      <Entities>
        <Entity>
          <Type> siemens.ad.bereichsarchiv:Dkl0007R3Lieferung</Type>
          <Properties>
            <Property Name="ixos.bai:LinkId">gss://$SIXXX0123456789$</Property>
            <Property Name="siemens.ad.bereichsarchiv:dokklasse">0007</Property>
            <Property Name="siemens.ad.bereichsarchiv:fk_sapsysid">GSSPEP</Property>
            <Property Name="siemens.ad.bereichsarchiv:fk_mandt">SIX</Property>
            <Property Name="siemens.ad.bereichsarchiv:lieferbelnr">0123456789</Property>
            <Property Name="siemens.ad.bereichsarchiv:lieferdat">9999-12-31</Property>
            <Property Name="siemens.ad.bereichsarchiv:fk_auftraggeber">GSSPEPS0</Property>
            <Property Name="siemens.ad.bereichsarchiv:erzeugungsdat">2008-07-18</Property>
          </Properties>
        </Entity>
      </Entities>
      <Condition>
        <and>
          <equal>
            <property name="siemens.ad.bereichsarchiv:lieferbelnr"/>
            <literal value="0123456789" datatype="String"/>
          </equal>
        </and>
      </Condition>
    </Command>
    <Command Type="Create" [DetailedResult="true|false"]>
      <Entities>
        <Entity>
          <Type> siemens.ad.bereichsarchiv:Document</Type>
          <Properties>
            <Property Name="ixos.dms:ArchiveId">4G</Property>
            <Property Name="siemens.ad.bereichsarchiv:sysname">GSS-Ausfuhrnachweis</Property>
            <Property Name="siemens.ad.bereichsarchiv:orgkrit1">SIX</Property>
            <Property Name="siemens.ad.bereichsarchiv:orgkrit2">A1234567</Property>
            <Property Name="siemens.ad.bereichsarchiv:orgkrit3">AuD</Property>
            <Property Name="siemens.ad.bereichsarchiv:fk_systyp">11</Property>
            <Property Name="siemens.ad.bereichsarchiv:fk_land">DE</Property>
            <Property Name="siemens.ad.bereichsarchiv:fk_modul">V</Property>
            <Property Name="siemens.ad.bereichsarchiv:fk_dokart">
              Ausfuhrnachweis_Papier
            </Property>
            <Property Name="ixos.bai:LinkedIds" MultiValue="true">
              <Value>gss://$SIXXX0123456789$</Value>
            </Property>
          </Properties>
          <Components>
            <Component>
              <Name>data</Name>
              <Format>application/pdf</Format>
              <Data><!-- Base64 kodierte Binärdaten --></Data>
            </Component>
          </Components>
        </Entity>
      </Entities>
    </Command>
  </Commands>
</LivelinkArchivingConnector>
```

21.3 Dialog „Livelink PDMS Connector Eigenschaften“

In diesem Dialog konfigurieren Sie die Zugangsdaten für die Kommunikation zwischen der inubit Process Engine und dem Livelink Archive Server.

Grundkonfiguration

- **Protokoll**
  Wählen Sie das HTTP oder HTTPS-Protokoll.
- **Servername**
  Domainname bzw. IP-Adresse des Ziel-Servers.
- **Port**
  Nummer des Ports für die Kommunikation zum Ziel-Server, z. B. xyz.inubit.com.
- **Domäne**
  Name der internen Domäne, an der Sie sich mit den Zugangsdaten „Benutzer“ und „Passwort“ anmelden möchten.
- **Benutzer/Passwort**
  Zugangsdaten, mit denen Sie sich an der oben genannten Domäne anmelden möchten.

22 Mail Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Mit einen Mail Connector können Sie E-Mails von einem POP3-, POP3S, IMAP- oder IMAPS-Server holen bzw. E-Mails über einen SMTP- oder SMTPS-Server versenden.

Konnektortypen

- **Input Connector**
  Holt E-Mails ab, bis das Postfach leer ist. Der Connector wird nur aktiv, wenn der Scheduler aktiviert ist.
- **Output Connector**
  Formatiert E-Mails im MIME-Format und versendet diese.

Stellen Sie sicher, dass der SMTP-Server von dem Rechner aus erreichbar ist, auf dem die inubit Process Engine installiert ist. Prüfen Sie, ob Mail-Relaying von der inubit Process Engine aus erforderlich ist und stimmen Sie die Konfiguration des SMTP-Servers entsprechend ab.
Siehe http://de.wikipedia.org/wiki/SMTP-Relay-Server.
Um den Mail Connector anonym als Mail-Relay zu nutzen, erfragen Sie die Relay-Konfiguration bei Ihrem Mail-Administrator. Bei dieser Nutzungsart können Sie die Verbindung nicht testen.

→ Siehe auch
- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Dialog „Zeitgesteuerte Verarbeitung“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1.1.3, S. 22)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

22.1 Modulvariablen des Mail Connectors

Bei der Ausführung eines Mail Connectors werden folgende Variablen gesetzt, die Sie auswerten oder überschreiben können:

- `Message-ID`

- `From` (nur Input Connector)
- `Reply-To` (nur Input Connector)
- `Subject` (nur Input Connector)

→ Für Infos über Variablen und deren Verwendung siehe *Workflow-Variablen und Mappings (Workbench: Benutzer-Guide, Kap. 14, S. 365)*.

22.2 Dialogbeschreibungen

Dieser Abschnitt erläutert die folgenden Themen:

22.2.1 Dialog „Mail Connector Eigenschaften“

(Input Connector)

In diesem Dialog legen Sie die Zugangsdaten zu dem Mailserver fest.

Grundeinstellungen

- **Protokoll**
  Protokoll, das der Konnektor verwenden soll.

Ihr Mailserver muss das gewählte Protokoll unterstützen!

  - **POP3**
    Übertragungsprotokoll für E-Mails, erlaubt das Auflisten, Abholen und Löschen von E-Mails auf dem Mailserver.
  - **POP3S**
    (POP über SSL/TLS) Die Verbindung zum Mailserver wird durch SSL verschlüsselt.

- **IMAP**
  Erweitertes Übertragungsprotokoll für E-Mails. Die E-Mails verbleiben in der Regel auf dem Mailserver, damit können E-Mails von verschiedenen Standorten aus zentral verwaltet werden.
- **IMAPS**
  (IMAP over SSL) Die Verbindung zum IMAP-Server wird beim Verbindungsaufbau durch SSL verschlüsselt.

- **Servername**
  Name oder IP-Adresse des Mailservers.
- **Port**
  Port, der für die Kommunikation genutzt wird. Der Button „Standard“ stellt den Standard-Port wieder her.

Authentifizierung

- **Konto**
  Benutzername für das Mailkonto.
- **Passwort**
  Passwort für das Mailkonto.

Zusätzliche Optionen

(nur bei IMAP und IMAPS als Protokoll)

AUTHENTICATE PLAIN Kommando bei der Authentifizierung vermeiden

Wenn markiert, wird die PLAIN-Authentifizierung deaktiviert.

Diese Option ist sinnvoll bei der Anbindung des Exchange-Servers 2007.

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

22.2.2 Dialog „Datenweitergabe konfigurieren“

(Input Connector)

In diesem Dialog haben Sie folgende Optionen:

Einstellungen für die Datenweitergabe

- **MIME**
  Für E-Mails im MIME-Format. Die E-Mail wird unverändert an das nächste Modul des Workflows weitergeleitet.
  Die MIME Spezifikation ermöglicht es, Nachrichten in Sprachen mit unterschiedlichen Zeichensätzen oder multimedial erweiterte Email auszutauschen. Eine MIME E-Mail enthält zusätzliche, standardisierte Felder in den Headern, welche die Datentypen des Inhalts und die Organisation der Nachrichten beschreiben.
- **MIME XML**
  Für E-Mails im MIME-Format. Die E-Mail wird entsprechend ihrem MIME Typ kodiert und der Inhalt der Email wird in das `IBIS-MIME.XML`-Format konvertiert.
  Für die Konvertierung wird folgendes Schema verwendet `Global>System>Mapping Templates>MIME Adapter> IBISMime1.1.xsd`
  - **Header-Namen im XML klein schreiben**
    Die Option ist sprechend. Einheitlich klein-geschriebene Namen vereinfachen den Zugriff auf die Elemente per XPath.
  - **Inhalte immer Base64 kodieren**
    Für den Transport von Binärdaten wie z. B. Grafiken.
- **Inhaltsverzeichnis des Postfaches als MIME XML**
  Lädt die Header-Informationen aus dem angegebenen Postfach herunter und konvertiert diese nach XML. Üblicherweise umfassen Header-Infos den Betreff (Subject), den Absender, das Datum und die Priorität.
  - **inkl. Inhalt der Nachrichten**
    - **Inhalte immer Base64 kodieren**

    Für den Transport von Binärdaten wie z. B. Grafiken.
  - **Header-Namen im XML klein schreiben**
    Die Option ist sprechend benannt. Die einheitliche Schreibung erleichtert den Zugriff auf die Daten via XPath.
- **Daten**
  - **aus dem Mitteilungsinhalt**
    Die Nachricht wird dem Message-Body entnommen, also dem Teil, der üblicherweise den Nachrichteninhalt enthält.
  - **aus dem Mitteilungsanhang**
    Die Nachricht wird dem Anhang entnommen. Beachten Sie, dass einige E-Mail-Programme (z. B. Outlook) auch plain-Text-Nachrichten als Anhang versenden!
    - **Anhangsname/oder die Nummer des Anhangsindex**:

    Wenn Sie keine Angaben machen, dann wird diejenige Datei im Workflow weitergeleitet, die als erstes an die E-Mail angehängt wurde. Auch wenn die E-Mail mehrere Anhänge hat, wird nur der erste weitergeleitet.

Um bei mehreren Anhängen gezielt einen bestimmten Anhang weiterzuleiten, geben Sie dessen Namen oder Position an.

22.2.3 Dialog „Mitteilungskonfiguration“

(Input Connector)

In diesem Dialog haben Sie folgende Optionen:

Nach dem Lesen löschen

Löschen

Wenn markiert, dann werden zuerst die Nachrichten vom Mailserver auf die inubit Process Engine kopiert und dann vom Mailserver gelöscht.

Filtereinstellungen

Zum Definieren von Filterkriterien, denen E-Mails entsprechen müssen, damit diese vom Mailserver abgeholt werden:

- **Verknüpfung der Filter**

  Wenn mehr als ein Filter angegeben ist:

  - UND

    Alle Filter müssen auf eine Nachricht zutreffen, damit diese geholt wird.

  - ODER

    Mindestens ein Filter muss auf eine Nachricht zutreffen, damit diese geholt wird.

- **Filter hinzufügen**

  Der Button fügt folgende Zeile ein:

  In dieser Zeile definieren Sie die Filterkriterien:

  - Subject/From/To: Liste zur Auswahl des Feldes, das gefiltert werden soll.
  - enthält/enthält nicht: Angabe, ob im gewählten Feld eine bestimmte Zeichenkette vorhanden sein muss oder nicht.
  - Eingabefeld: Zur Eingabe der Zeichenkette, die vorhanden oder nicht vorhanden sein soll.

22.2.4 Dialog „S/MIME Entschlüsselung“

(Input Connector)

In diesem Dialog haben Sie folgende Optionen:

- **Entschlüsselung aktivieren**
  Aktivieren Sie diese Option, wenn die E-Mails, die der Mail Connector abholen soll, verschlüsselt sind.

Privaten Schlüssel hinzufügen:

- **Hinzufügen (Button)**
  Der Button öffnet einen Dateiexplorer zum Laden des privaten Schlüssels, mit dem die E-Mails entschlüsselt werden.

Zum Laden benötigen Sie den Alias und das Passwort des Schlüssels!

Öffentliche Daten des Schlüssels

Zeigt die öffentlichen Daten des Zertifikats, nachdem der private Schlüssel geladen wurde.

22.2.5 Dialog „Eigenschaften der Eingabedaten“

(Output Connector)

In diesem Dialog haben Sie folgende Optionen:

Einstellungen für das Versenden

- **MIME formatiert (RCF 822)**
  Zum Senden von Mime-formatierten Eingangsnachrichten.
- **MIME XML**
  Zum Senden von Eingangsnachrichten, die entsprechend der XML-Schemadatei unter `Global > System > Mapping Templates > MIME Adapter > IBISMime1.1.xsd` formatiert sind.

Wenn Sie „MIME formatiert“ oder „MIME XML“ gewählt haben, dann werden die Angaben in den Eingangsnachrichten nur genutzt, wenn Sie den Bereich „Nachricht“ im *Dialog „SMTP Connector Eigenschaften“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 22.2.6, S. 222)* leer lassen.

- **Daten**
  Zum Senden von Nachrichten in einem beliebigen Format:
  - **MIME-Typ der Daten**

Typangabe der übermittelten Daten.

- **Content-Transfer-Encoding**

  Zum Festlegen einer sicheren Kodierung für nicht-ASCII-Zeichen (z. B. Sonderzeichen) und nicht-Textteile (z. B. Binaries) bei der Übertragung. Stellt den verlustfreien Datenaustausch sicher.

  - **auto**:

    Der Wert des Headers „Content-Transfer-Encoding“ wird automatisch entsprechend dem Content-Type und den zu versendenden Daten festgelegt.

  - **base64**:

    Zum Übermitteln z. B. von binären oder 8-bit-kodierten Nachrichten.

- **Als E-Mail-Anhang**

  Sendet die Eingangsnachricht als Anhang einer E-Mail.

  - **Name des Anhangs**

    Die Option ist sprechend benannt.

    Sie können an den Namen des Anhangs das Datum, den Zeitstempel oder die Prozess-ID anhängen:

    Datum:
    Sie können das Datum unterschiedlich formatiert ausgeben:
    - Als Angabe ohne Datumsformat wird aus `name_%TimeStamp%%.xml` die Angabe `name_06082008-151047.xml` erzeugt, dabei ist name= Anhangname, 12122002 = Datum im Format `ddMMyyyy`, 151047= Zeit im Format `hhmmss`.
    - Als Angabe mit Datumsformat `hh` wird aus `name_%TimeStamp%hh%.xml` die Angabe `Name_11.xml` erzeugt.
    - Bei der Angabe mit Datumsformat `ddMM` wird aus `Name_%TimeStamp%ddMM%.xml` die Angabe `Name_2309.xml` erzeugt.

    Zeitstempel:
    Der Zeitstempel enthält zusätzlich eine Angabe in Millisekunden. Die Angabe `name_%TimeStamp%ddMMyyyy-HHmmss-SSS%.xml` erzeugt `name_12122002-151304-027.xml`.

    Prozess-ID:
    Um die Prozess-ID auszugeben, geben Sie den Anhangnamen im folgenden Format an: `name_PID_%ProcessId%.xml`.

  - **zusätzlichen Textanhang eingeben**

    Sie können zusätzlich Text eingeben, der als Anhang übermittelt wird.

22.2.6 Dialog „SMTP Connector Eigenschaften“

(Output Connector)

In diesem Dialog legen Sie die Zugangsdaten für den SMTP-Server und Eigenschaften der Nachrichten fest.

Grundeinstellungen

- **Protokoll**
  - **SMTP**: Protokoll zum Senden und Weiterleiten von E-Mails.
  - **SMTPS:** (SMTP über SSL) Verschlüsselt die Verbindung mit SSL.
- **Servername**
  URL des SMTP-Servers (Domainname oder IP-Adresse). Wenn Sie keine Angabe machen, dann wird automatisch „localhost“ gesetzt.
- **Portnummer**
  Voreingestellt auf 25 bzw. 465. Der Button „Standard“ stellt die Voreinstellung nach Änderungen wieder her.
- **Authentifizierung**
  Zu markieren, wenn der Mailserver eine Authentifizierung verlangt.
  - **Konto**: Benutzername.
  - **Passwort**: Passwort, dass Sie zusammen mit dem Benutzernamen erhalten haben.

Nachricht

Sie können die Felder leer lassen, wenn die Eingangsnachricht als „MIME formatiert“ oder „MIME-XML“ versendet wird und bereits Nachrichteneigenschaften wie Absender, Empfänger, Betreff) enthält.

Aus den MIME-Headern, die in der Eingangsnachricht bereits vorhanden sind, wird eine Liste der Empfänger mit der Reihenfolge TO, CC, BCC erstellt. Für jeden Empfänger wird ein `RCPT TO:`-Kommando an den Mailserver geschickt.

Wenn Sie die Felder im Bereich „Nachricht“ füllen, dann überschreiben diese Angaben die Werte aus der Eingangsnachricht!

- **Von**: Absenderadresse.
- **An**: Empfängeradresse. Mehrere Empfängeradressen werden durch Kommata getrennt.
- **Cc**: Weitere Empfängeradressen. Mehrere Empfängeradressen werden durch Kommata getrennt.
- **Bcc**: Empfängeradresse an, die für To- und Cc-Empfänger nicht sichtbar ist. Mehrere Empfängeradressen werden durch Kommata getrennt.

- **Rückantwort an**: Auszufüllen, wenn die Absenderadresse nicht die Adresse ist, unter der Sie eine Rückantwort erhalten wollen.
- **Betreff**: Gibt an, worauf die Nachricht Bezug nimmt.

Sie können für den Betreff Datum, Zeitstempel oder Prozess-ID hinzufügen lassen:
- **Datum**:
Sie können das Datum unterschiedlich ausgeben lassen. Als Angabe ohne Datumsformat wird aus `betreff_%TimeStamp%%.xml` die Angabe `betreff_06082008-151047.xml` erzeugt. Als Angabe mit Datumsformat `hh` wird aus `betreff_%TimeStamp%hh%.xml` die Angabe `Betreff_11.xml` erzeugt. Bei der Angabe mit Datumsformat `ddMM` wird aus `Betreff_%TimeStamp%ddMM%.xml` die Angabe `Betreff_2309.xml` erzeugt.
- **Zeitstempel**:
Der Zeitstempel enthält zusätzlich eine Angabe in Millisekunden. Die Angabe `betreff_%TimeStamp%ddMMyyyy-HHmmss-SSS%.xml` erzeugt `betreff_12122002-151304-027.xml`.
- **Prozess-ID**:
Um die Prozess-ID mit auszugeben, geben Sie den Betreff im folgenden Format an: `betreff_PID_%ProcessId%.xml`.

Verbindungstest

- **Verbindung testen**
Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

22.2.7 Dialog „S/MIME Verschlüsselung“

(Output Connector)

In diesem Dialog können Sie Nachrichten S/MIME-verschlüsseln.

- **Verschlüsselung aktivieren**
Die Option ist sprechend benannt.

Zertifikat des Empfängers hinzufügen

- **Hinzufügen**
Der Button öffnet einen Dateiexplorer, um die Keystore-Datei mit dem Schlüssel zu laden. Zum Laden benötigen Sie den Alias des Zertifikats.

Daten des Zertifikats	Zeigt nach dem Laden die öffentlichen Daten des Zertifikats an.

23 MSMQ Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Microsoft Message Queuing (MSMQ) ist die Implementation einer Nachrichten-Warteschlange u. a. zur Anbindung von MS Navision, MS Axapta, MS CRM und MS Biztalk.

MSMQ entkoppelt Sender- und Empfängeranwendungen, so dass die Senderanwendung Nachrichten senden kann und sich nicht darum kümmern muss, ob und wann diese an die Empfängeranwendung ausgeliefert werden. Alle Nachrichten werden von MSMQ in einer Warteschlange gespeichert und weitergeleitet, sobald die Empfängeranwendung erreichbar ist.

Nachrichten werden also auch dann zuverlässig übermittelt, wenn z. B. die Empfängeranwendung nicht läuft oder der Rechner, auf dem die Empfängeranwendung installiert ist, vorübergehend nicht mit dem Netzwerk verbunden ist.

Konnektortypen

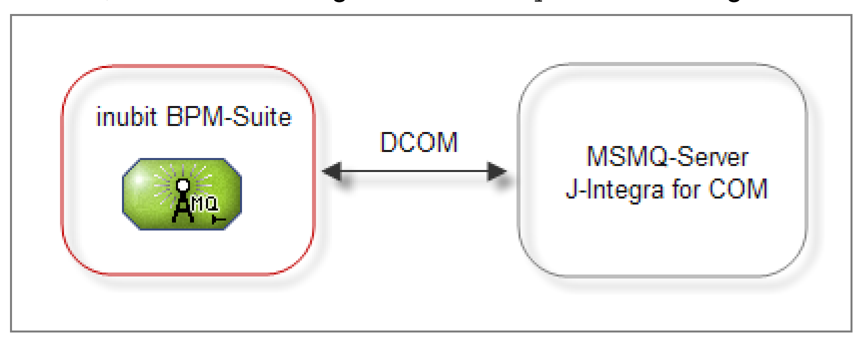
- **Input Listener Connector**
  Fungiert als Empfänger und wartet auf Nachrichten aus der konfigurierten Warteschlange.
- **Medium/Output Connector**
  Agiert als Sender und schreibt Nachrichten in die angegebene Warteschlange. Beide Konnektortypen geben die Eingangsnachricht wieder aus.

→ Siehe auch
  - *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
  - *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
  - *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

23.1 Voraussetzungen

- Zugriff auf einen MSMQ-Server
- Installation von J-Integra® for COM

J-Integra® for COM muss auf dem MSMQ-Server installiert werden, weil dort die nötige Bibliothek `mqoa.dll` vorliegt:



Eine detaillierte Installationsanleitung finden Sie unter http://j-integra.intrinsyc.com/support/com/doc/.

- Installation eines DCOM-Surrogats

  Um über DCOM auf die MSMQ-COM-Komponente zugreifen zu können, muss das DCOM-Surrogat mit dem Befehl `setdllhost.exe mqoa.dll "MSMQ"` auf dem MSMQ-Server installiert werden. Die Bibliothek `mqoa.dll` ist Teil des MSMQ-Servers.

- Java-Wrapper `msmq.jar`

  Für den Zugriff auf die Bibliothek `mqoa.dll` aus der inubit Suite 6.

  → Siehe *Treiber installieren (Process Engine: Administrator- und Entwickler-Guide, Kap. 3.4, S. 51).*

Sie erhalten die Software und den Java-Wrapper zusammen mit der Lizenz für den MSMQ Connector von der inubit AG.

23.2 Ein- und Ausgangsnachrichten erstellen

Die Ein- und Ausgangsnachrichten erstellen Sie mit Hilfe eines Templates und eines XSLT Converters.

So gehen Sie vor

1. Erstellen Sie einen XSLT Converter.
2. Öffnen Sie im Bereich „XML-Zieldatei“ das ⬇ -Menü.
3. Wählen Sie „Öffnen von > Repository". Der Repository Explorer öffnet sich.
4. Öffnen Sie das Verzeichnis `Global > System > Mapping Templates > MSMQ Connector`.
5. Markieren Sie eine der folgenden Dateien:
   - Eingangsnachricht erstellen: `MSMQ_Input.xsd`

- Ausgangsnachricht erstellen: `MSMQ_Output.xsd`

6. Klicken Sie auf „OK“. Der Explorer schließt sich und das Template wird angezeigt.
7. Bilden Sie Ihre Eingangsnachricht auf die Template-Struktur ab, um eine Nachricht im erforderlichen Format zu erzeugen.
   → Siehe *XSLT Converter (Data Converter) (Workbench/Process Engine: Modul-Guide, Kap. 5, S. 133).*

23.3 Dialog „MSMQ Connector Eigenschaften“

In diesem Dialog haben Sie folgende Optionen:

Einstellungen

- **Server**
  IP-Adresse oder Hostname des MSMQ-Servers, zu dem die Verbindung aufgebaut werden soll.
- **Warteschlange**
  Name der Warteschlange auf dem MSMQ-Server.

Authentifizierung

- **Domäne**
  Name der Verwaltungsstruktur im Windows-Netzwerk, an der Sie sich mit den Zugangsdaten aus den Feldern „Benutzer"/ “Passwort" anmelden möchten.
- **Benutzer/Passwort**
  Zugangsdaten des Benutzers, mit dem Sie sich an der oben genannten Domäne anmelden möchten. Der Benutzer muss berechtigt sein, auf die Warteschlange zuzugreifen.

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

24 OFTP Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Der OFTP Connector wird genutzt, um Geschäftsdaten gemäß dem Odette File Transfer Protocol (OFTP) auszutauschen.

OFTP bietet die Möglichkeit, eine Verbindung nach dem Abbruch wieder aufzusetzen, ohne die gesamte Datei erneut zu übertragen. Es können ASCII- und EBCDIC-Dateien übertragen werden.

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

24.1 Voraussetzungen

- rvs Software der Firma gedas Deutschland GmbH muss installiert und konfiguriert sein.

  Dabei müssen Übertragungsparameter eingestellt und Stationen mit einer definierten Station-ID eingerichtet werden. Die Software besteht aus der rvs Middleware, dem rvs Server und einer Client-Komponente und kann sowohl über das Benutzerinterface der Middleware als auch über das Benutzerinterface des rvs Servers konfiguriert werden.

Erfragen Sie unbedingt bei inubit AG, welche rvs-Version Sie für Ihre Installation benötigen.

- `rvsclient.jar`

Die Datei `rvs client/classes/rvsclient.jar` der rvs Client-Komponente muss als Treiber-Bibliothek auf die inubit Process Engine hochgeladen werden.

→ Siehe *Treiber installieren (Process Engine: Administrator- und Entwickler-Guide, Kap. 3.4, S. 51).*

- OFTP als Listener betreiben

  Um den OFTP Connector im Listener Modus zu betreiben, müssen Sie in der rvs Software auf ein Skript verweisen. Dieses Skript sendet eine Benachrichtigung an den OFTP Connector, sobald eine Geschäftsnachricht eingetroffen ist. Das Skript ist im Lieferumfang der inubit Suite 6 enthalten.

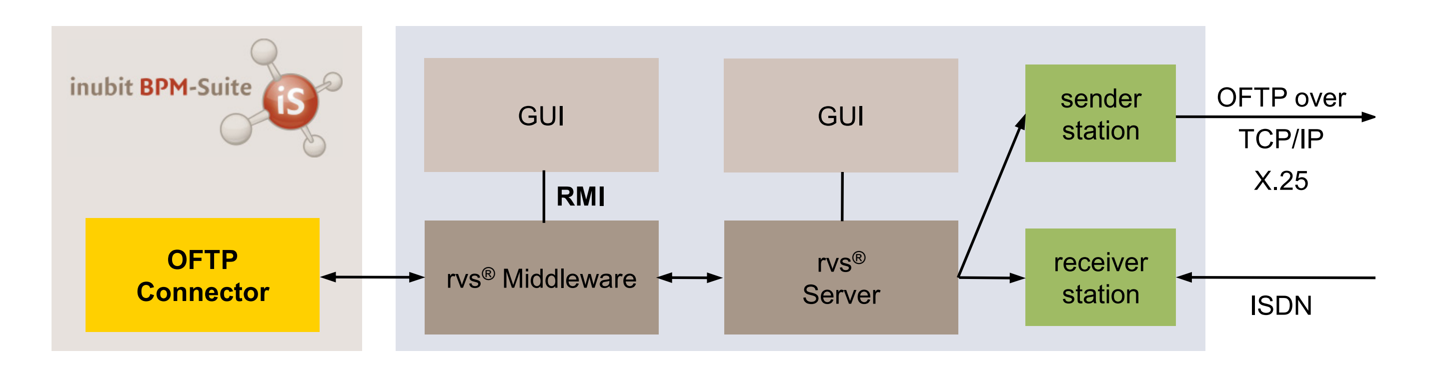
  a. Legen Sie in der GUI der rvs-Middleware einen Empfangs-Job an (residenter Empfangseintrag).

  b. Geben Sie das Skript als zu startenden Job an. Das Skript liegt im Verzeichnis `<iS-installdir>/server/ibis_root/conf/bin`.

     - AIX: `rvs_is_receive.pl`.
     - Windows: `rvs_is_receive.bat`.

Für weitere Informationen beachten Sie die Dokumentation der rvs Software.

24.2 Funktionsprinzip



Nachrichtenfluss

- **Nachricht versenden mit Output Connector**

  Der OFTP Connector übergibt die Nachrichten noch auf der inubit Process Engine an rvs. Der Versand kann mit Hilfe des Schedulers so konfiguriert werden, dass die Nachricht sofort

verschickt wird, wenn sie im Workflow am OFTP Connector ankommt, oder nur zu einer vereinbarten Zeit oder in einem Zeitintervall.

rvs verschickt die Nachricht über das vereinbarte Protokoll an die vom Empfänger eingerichtete Station.

- **Nachricht empfangen**
  - **Input Connector**
    Der OFTP Connector prüft aktiv in einem festgelegten Zeitintervall, ob Nachrichten vorhanden sind.
  - **Input Listener Connector**
    Der OFTP Connector wartet auf der konfigurierten Verbindungen auf das Eintreffen von Nachrichten.

Abhängig davon, wie Ihr Connector konfiguriert ist, findet die Kommunikation synchron oder asynchron statt, siehe *Datenübertragungsmodi (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 24.3, S. 231)*.

24.3 Datenübertragungsmodi

Der OFTP Connector unterstützt folgende Datenübertragungsmodi:

- **Asynchrone Übertragung mit Input Connector**

  Der Input Connector holt die Daten entsprechend der Konfiguration im „*Dialog „Zeitgesteuerte Verarbeitung“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1.1.3, S. 22)*“.

  rvs sendet eine Bestätigung (EERP), wenn die Verbindung weiter besteht oder beim nächsten Verbindungsaufbau. Dies entspricht der Option „Normal“ für den Parameter EERP_out.

  Das Löschen beendeter Jobs (Nachrichtenempfang) muss unterdrückt werden (Einstellung „setparm CMDDELETE=0“ im rvs).

  → Siehe Parameters EERP_out im Abschnitt *Nachbarstationen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 24.6.1, S. 237)*.

Die Bestätigung garantiert dem Absender der Nachricht nicht, dass die Nachricht auf der inubit Process Engine angekommen ist! Achten Sie darauf, dass die zulässige Größe der Datenbank im Verlauf des Betriebs nicht überschritten wird.

Wenn Sie mehr Informationen als den Dateinamen erhalten möchten, dann müssen Sie die rvs-Konfiguration entsprechend ändern.

- **Synchrone Übertragung durch rvs mit Input Listener Connector**

  rvs bestätigt per EERP-out den Empfang der Nachricht.

  Dieses Verfahren ist in Bezug auf den OFTP Connector halbsynchron, weil die Bestätigung versendet wird, bevor der OFTP Connector bestätigen kann, dass die Nachricht auf der inubit Process Engine vorliegt.

  Das Löschen beendeter Jobs (Nachrichtenempfang) muss unterdrückt werden (Einstellung „setparm CMDDELETE=0“ im rvs).

- **Synchrone Übertragung mit Input Listener Connector**

  Die Empfangsbestätigung wird verschickt, wenn sie vollständig auf der inubit Process Engine eingetroffen ist.

  In der Konfiguration der Nachbarstation muss der Parameter EERP_out auf „HOLD/IMMEDIATE“ gesetzt werden.

  → Siehe Parameter EERP_out im Abschnitt *Nachbarstationen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 24.6.1, S. 237)*.

  Der Job (Nachrichtenempfang) bleibt aktiv, bis die Empfangsbestätigung (EERP_out) gesendet wurde.

- **Synchrone Übertragung mit Output Connector**

  Setzen Sie beim Konfigurieren des Connectors ein Timeout, damit im Fehlerfall keine Blockade durch das Warten entsteht. Berücksichtigen Sie bei dem Wert die Nachrichtengröße und den Übertragungsweg.

  Kontrollieren Sie die Konfiguration der Gegenstelle: wenn dort die EERP-Nachricht auf „hold“ steht, kann es auch zu einer Blockade kommen.

  → Siehe *OFTP als Listener betreiben (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 24, S. 230)*

  Send Job bei erfolgreichem/fehlgeschlagenem Senden: `rvs_is_send`.

24.4 rvs Monitoring

Mit dem Kommandozeilentool „StartCLI“ können Sie u. a. die Verbindung der inubit Process Engine mit dem rvs Server prüfen.

Optionen

Option	Information
`-r,--rvs`	Verbindung der inubit Process Engine mit dem rvs Server prüfen
`--rhost`	rvs Host
`--rlogin`	rvs Login
`--rmwname`	Name der rvs Middleware
`--rpassword`	rvs Passwort

→ Siehe *Remote-Überwachung mit Command Line Interface (StartCLI) (Process Engine: Administrator- und Entwickler-Guide, Kap. 7.10, S. 108).*

24.5 Dialog „OFTP Datenaustausch-Konfiguration“

Der Dialog zur Datenaustausch-Konfiguration unterscheidet sich für die verschiedenen Konnektortypen.

Grundeinstellungen

- **Middleware-Serveradresse**: Geben Sie hier die Rechnernamen oder die IP-Nummer an, beispielsweise „MyServer“ oder „123.456.7.8“. Bei der Eingabe müssen Sie die Anführungsstriche weg lassen. Wenn die RMI Registry nicht auf Port 1099 läuft, geben Sie die Portnummer an, die mit einem Doppelpunkt vom Rechnernamen getrennt wird, beispielsweise „MyServer:1234“.
- **Middleware-Name**: Hier wird üblicherweise die Zeichenkette „rvsmw“ benutzt. Wenn Sie einen anderen Namen für die Middleware vergeben haben, übernehmen Sie diesen hier.
- **Anmeldung**: Geben Sie hier die Benutzernamen an, mit dem Sie sich bei der Middleware anmelden. Groß- und Kleinschreibung wird unterschieden.
- **Passwort**: Geben Sie hier Ihr Passwort ein.
- **Dateiname**

  Der Name wird für die OFTP Übertragung benutzt und muss dem Empfänger bekannt gemacht werden, damit dieser auf die Datei zugreifen kann.

  Im Dateinamen können Sie Wildcards oder reguläre Ausdrücke verwenden:

  - **Wildcards**
    - Fragezeichen (?): Steht für genau ein beliebiges Zeichen.

- Asterisk (*): Steht für Null bis n beliebige Zeichen.

Die Wildcards können in beliebiger Kombination und Häufigkeit benutzt werden.

Wenn Wildcards angegeben sind, dann holt z. B. ein Input Connector nur Dateien, die dem angegebenen Wert entsprechen, alle anderen Dateien werden ignoriert. Um alle Dateien zu empfangen, geben Sie * ein.

- **Reguläre Ausdrücke**

Reguläre Ausdrücke müssen in Schrägstriche eingeschlossen werden, z. B. `/(\d\d).(\d\d).(\d\d\d\d)/`.

Für Informationen über reguläre Ausdrücke siehe z. B. http://www.perl.com/doc/manual/html/pod/perlre.html oder http://www.regular-expressions.info/tutorial.html.

Windows: Das Windows Betriebssystem ermöglicht es, Dateien auszublenden. Wenn ausgeblendete Dateien einem Wildcard-Kriterium genügen, werden sie auch übertragen. Stellen Sie sicher, dass die Dateien, die Sie im Verzeichnis des Explorers sehen können, auch alle Dateien sind, die hier enthalten sind. Blenden Sie dazu alle Dateien ein (im Explorer Optionen „Extras“ - „Ordneroptionen“).

- **Inbox-Unterverzeichnis**: Geben Sie den Verzeichnisnamen an, den Sie in der Middleware unter „Inbox“ angelegt haben und in das die eingehenden Dateien abgelegt werden sollen.

  Der Ordner `Inbox` ist identisch mit dem Ordner `<rvs-install-dir>/usrdat` auf dem rvs Server. Wenn Sie kein anderes Verzeichnis konfiguriert haben, werden hier die eingehenden Nachrichten abgelegt.
- **Virtueller Dateiname** (beim Input Connector nicht vorhanden): Sie können im virtuellen Dateinamen (virtual dataset name = VDSN) als erstes einen Verzeichnisnamen angeben, beispielsweise `abc/123order.edi` - in diesem Fall wird die Datei bei Ihrem Geschäftspartner in folgendes Verzeichnis gelegt `<rvs-install-dir>/usrdat/inbox/abc/`
- **Empfängerstations-ID** (beim Input Connector nicht vorhanden): Eine Stations-ID (SID) kann bis zu 16 Zeichen umfassen. Diese ID wird nur lokal genutzt. Die ID ist frei wählbar und kann aus Zahlen und Buchstaben bestehen. Diese SID muss zuerst in der rvs Software angelegt werden und kann dann hier übernommen werden.
- **Stationen**

  Sie können den Dialog „Stationen“ nur aufrufen, wenn Sie zuvor gültige Verbindungsdaten zu einer OFTP Middleware im Dialogabschnitt „Grundeinstellungen“ eingegeben haben.

- **Textdatei**: Wählen Sie diese Option, wenn der OFTP Connector auf einem anderen Betriebssystem ausgeführt wird als die Middleware. Mit dieser Option wird eine automatische Konvertierung aktiviert, welche die unterschiedlichen Betriebssysteme erkennt und die benötigten Steuerzeichen entsprechend setzt. Windows-Dateien z. B. haben am Ende jeder Zeile ein Carriage Return-Zeichen und ein Line Feed-Zeichen, während bei UNIX Systemen am Ende der Zeile nur ein Line Feed-Zeichen steht.

Deaktivieren Sie diese Option zum Übertragen binärer Dateien!

- **Datei auf dem Server löschen**: Nach der Übertragung wird die Datei auf dem rvs Server gelöscht, wenn diese Option ausgewählt ist.
- **Synchron** (nur beim Output-Connector)
  Wenn die Checkbox markiert ist, wird der synchrone Datenübertragungsmodus aktiviert.
- **RVS Evo**
  Markieren Sie diese Option, wenn Sie die rvs Software RVS Evolution (statt der Classic-Version) verwenden.

Format

(nur bei Output Connector)

Geben Sie hier die Formatinformation für die Datei an, die Sie übertragen wollen.

- **Text**: Textdateien im ASCII Format
- **Feste Sätze**: Dateien, die feste Satzlängen enthalten.
- **Variable Sätze**: Dateien, deren Satzlängen variabel sind
- **Unstrukturiert**: Wählen Sie diese Option, wenn Sie binäre Dateien übertragen wollen

VFTyp

(nur bei Output Connector)

Mit dem VFTyp bestimmen Sie, wie die Dateien vor der Übertragung konvertiert werden sollen. Die Konvertierung ist nur auf Dateien mit festem oder variablem Format anwendbar (siehe Option „Format“ über dieser Option). Sie können zwischen den Optionen Text, Variable oder Speziell wählen.

- **Text**: Die Datei wird nicht konvertiert. Die Datei muss bereits dem Format entsprechen und über die Satzlängen verfügen, die vom Geschäftspartner erwartet werden.
- **Variable**: Wenn Sie diese Option wählen, wird die Datei auf Grundlage eines in rvs angegebenen Standardwertes konvertiert.

- **Speziell**: Mit dieser Option wird die Datei in ein spezielles Format gewandelt, das für die Übertragung im FT-SINIX Modus gebraucht wird (**F**ile **T**ransfer **Si**emens U**NIX**)

Code In/Code Out

(nur bei Output Connector)

Mit den Optionen „Code In“ und „Code out“ können Sie die Nachricht so konvertieren, dass sie mit dem Zeichensatz des Quell- respektive Zielbetriebssystems übereinstimmt. Es gibt jeweils die Möglichkeit, die Nachricht in den Zeichensatz ASCII oder EBCDIC zu konvertieren. Der Zeichensatz ASCII wird bei Windows und UNIX Betriebssystemen eingesetzt. EBCDIC ist der Standard für OS/400 und OS/390.

→ Siehe
- *Dialog „Stationen“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 24.6, S. 236)*
- *Datenübertragungsmodi (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 24.3, S. 231)*

24.6 Dialog „Stationen“

Dieser Abschnitt erläutert die folgenden Themen:

Der Dialog „Stationen“ lässt sich nur aufrufen, wenn zuvor gültige Verbindungsdaten zu einer OFTP Middleware im Dialogabschnitt „Grundeinstellungen“ eingegeben wurden. Wenn Sie den „Stationen“ Dialog aufrufen, verbindet sich der OFTP Connector über die von Ihnen angegebenen „Grundeinstellungen“ mit der OFTP Middleware und erhält dort die Konfigurationsdaten, die im „Stationen“ Dialog angezeigt werden.

Beachten Sie, dass Einstellungen im Dialog „Stationen“ die Konfiguration in der OFTP Middleware ändern, wenn Sie den Dialog mit „OK“ beenden.

LOC

„LOC“ ist die lokale Stationsdatei. Wenn Sie diesen Eintrag auswählen, können Sie im rechten Fenster die Konfiguration der lokalen Station angeben. Die Konfigurationsdaten müssen mit denen übereinstimmen, die Sie in Ihrer rvs-Administrator-Konsole angegeben haben.

Die lokale Station muss immer den Namen „LOC“ (in Großbuchstaben) haben.

Die ersten beiden Optionsgruppen sind die folgenden:

- **Odette ID**: Dies ist eine eindeutige Zeichenkette, die aus maximal 25 Zeichen besteht. Das erste Zeichen ist immer ein „O“.

  Eine Odette ID erhalten Sie unter www.vda.de

- **Lokale Station**: Name, Telefon und Bemerkungen sind optionale Felder. Vergeben Sie für „Name“ einen beschreibenden Namen; das Feld Telefon kann die Nummer des Sachbearbeiters enthalten, der die Konfiguration betreut; „Bemerkungen“ kann den frei wählbaren Namen der Station enthalten.

Danach folgen die Empfänger. Deren Konfiguration ist abhängig vom gewählten Protokoll. Wenn Sie „LOC“ markiert haben, steht Ihnen ein Kontextmenü mit folgenden Befehlen zur Verfügung:

- Nachbarstationen hinzufügen
- Empfänger hinzufügen
- Empfänger entfernen
- Verbindung aktivieren

24.6.1 Nachbarstationen

Dieser Abschnitt erläutert die folgenden Themen:

Eine Nachbarstation ist die Station eines Geschäftspartners, mit dem Sie Nachrichten austauschen wollen.

Nachbarstation hinzufügen

Mit dieser Option fügen Sie eine neue Nachbarstation hinzu. Diese wird in einem Verzeichnisbaum unter „LOC“ hinzugefügt. Sie müssen die Konfiguration der Nachbarstation kennen. Wenn Sie die neu hinzugefügte Station auswählen, können Sie die Parameter auf der

rechten Seite des Dialogs hinzufügen. Die Stationen können für verschiedene Übertragungsprotokolle konfiguriert werden. Wählen Sie eines der Protokolle: X.25, lu6.2, TCP/IP, ISDN oder Virtual.

Dialoggruppe Odette

- **Odette ID**: Dies ist eine eindeutige Zeichenkette, die aus maximal 25 Zeichen besteht. Das erste Zeichen ist immer ein „O“.
- **Passwort empfangen**: Das vom Geschäftspartner erwartete Passwort.
- **Passwort senden**: Das an den Geschäftspartner gesendete Passwort.
- **Größe des Austauschpuffers**: Die Größe wird in Byte angegeben. Der Standard ist 0. Wenn Sie hier einen Wert setzen wollen, benutzen Sie den Wert des globalen Parameters OEXBUF, wie im Handbuch der verwendeten OFTP Middleware beschrieben.
- **Anzahl der Austauschpuffer**: Der Standard ist 0. Die Zahl bestimmt die Anzahl der Puffer die gesendet werden, ohne auf eine Antwort zu warten.
- **Empfangene Blockanzahl**: Da OFTP Nachrichten bei einer Verbindungsstörung nicht vollständig neu gesendet werden, sondern nur von sogenannten Aufsetzpunkten, kann hier bestimmt werden, nach wie viel empfangenen Blöcken ein Aufsetzpunkt erstellt wird. Die Anzahl der Aufsetzpunkte hat Einfluss auf Performance und Speicherverbrauch, benutzen Sie daher für stabile Verbindungen eine kleine Zahl und für störanfällige Verbindungen eine große. Der Standard ist 0. Wenn Sie hier einen Wert setzen wollen, benutzen Sie den Wert des globalen Parameters RECVBLOCKS, wie im Handbuch der verwendeten OFTP Middleware beschrieben.
- **Gesendete Blockanzahl**: Die Bedeutung ist gleich der in der Option „Empfangene Blockanzahl“. Der globale Parameter heißt hier SENDBLOCKS.
- **Kompression**: Diese Option ist eine Combobox. Sie können zwischen drei Modi wählen.
  - **ODETTE**: Die Blöcke werden an die OFTP Middleware übertragen und dort nach dem Odette Verfahren komprimiert.
  - **NONE**: Die Datenblöcke werden nicht komprimiert.
  - **GNU**: Die Datenblöcke werden bereits in der inubit Process Engine mit GNU ZIP (gzip) komprimiert und dann an die OFTP Middleware übergeben.
- **VDSN Zeichenmenge**: Diese Option ist eine Combobox. Sie können unter verschiedenen Zeichensätzen wählen:
  - **ALL**: Es gibt keine Einschränkungen bei der Zeichenmenge.
  - **ODETTE**: alle Großbuchstaben, Ziffern und die Sonderzeichen () -. / &

- **CHECK_RE**: Erlaubt wie bei ALL alle Zeichen, Voraussetzung ist allerdings das ein RE besteht. Dieser wird dann ausgeführt. **RE** steht für „Residenter **E**mpfangseintrag“. Ein residenter Empfangseintrag wird für eine Datei mit einem bestimmten Namen angelegt. Trifft eine Datei mit diesem Namen ein, so bestimmt der RE welche Aktion danach gestartet werden soll, beispielsweise Datei ersetzen, Datei löschen oder, wie hier gefordert, Code konvertieren. Für Residente Empfangseinträge gibt es in der OFTP Middleware eine eigene GUI, zu deren Definition.
- **OFTPUNIX**: Alle Großbuchstaben, Ziffern und die Sonderzeichen `.` `-` (Punkt, Minus).
- **UNIX**: alle Buchstaben, Ziffern und die Sonderzeichen `#_-+.`

■ **Eingangsbestätigung (EERP_in)**: EERP (End to End Response) bedeutet Empfangsbestätigung. Die EERP_in wird vom Geschäftspartner gesendet. Die Combobox bietet die Einträge NEVER und NORMAL.

- **NEVER**: bedeutet, dass der Geschäftspartner keine Eingangsbestätigung sendet. Die Übertragung wird mit der korrekten Übermittlung der Nachricht beendet.
- **NORMAL**: bedeutet, dass eine Eingangsbestätigung erwartet wird. Der Eingang der Bestätigung gehört zur Sendeaufforderung. Eine Sendeaufforderung besteht so lange wie entweder noch Dateien zur Übertragung anstehen oder auf den Eingang von Eingangsbestätigungen gewartet wird. Mit dem Eingang der Bestätigung wird auch die Sendeaufforderung beendet. Die Auswahl NORMAL ist der Standard.

■ **Ausgangsbestätigung (EERP_out)**: Eine Empfangsbestätigung wird von der lokalen Station an den Geschäftspartner gesendet. Standardmäßig ist die Option IMMEDIATE vorausgewählt. Die folgenden Möglichkeiten stehen zur Verfügung.

- **NEVER**: Es wird keine Empfangsbestätigung gesendet.
- **NORMAL**: Beim Empfang einer Nachricht wird eine Empfangsbestätigung gesendet. Wenn die Verbindung noch besteht, wird die Bestätigung sofort gesendet. Wenn die Verbindung bereits abgebaut wurde, wird die Bestätigung beim nächsten Verbindungsaufbau gesendet.
- **SYNC**: Die Empfangsbestätigung wird über dieselbe Verbindung gesendet, über welche die Nachricht eingegangen ist. Das Bestehen der Verbindung wird dabei erzwungen. Es wird geprüft, ob die Empfangsbestätigung erfolgreich versendet wurde, erst dann kann die Verbindung geschlossen werden.
- **IMMEDIATE**: Die Empfangsbestätigung wird sofort gesendet. Sollte die Verbindung bereits beendet worden sein, wird sie für die Empfangsbestätigung erneut aufgebaut.

- **HOLD**: Die Empfangsbestätigung wird erst gesendet, wenn sie explizit von einem Operator in der OFTP Middleware freigegeben wurde. Besteht die Verbindung noch, so wird die Empfangsbestätigung sofort gesendet. Wenn die aktuelle Verbindung bereits abgebaut wurde, wird die Empfangsbestätigung gesendet wenn das nächste Mal eine Verbindung erstellt wurde.

Der OFTP Connector erteilt die Freigabe für die OFTP Middleware selbständig, es ist keine Interaktion eines Operators erforderlich.

- **HOLD/IMMEDIATE**: Für die Empfangsbestätigung wird auf einen Freigabe gewartet. Wenn diese erteilt wurde, wird die Empfangsbestätigung sofort über die noch bestehende Verbindung gesendet. Wurde diese bereits beendet, wird sofort eine neue Verbindung für das Versenden der Empfangsbestätigung aufgebaut und die Bestätigung wird versendet.

Der OFTP Connector erteilt die Freigabe für die OFTP Middleware selbständig, es ist keine Interaktion eines Operators erforderlich.

Neighbour Station

- Name, Telefon, Bemerkungen und Netzwerk sind optionale Felder. Geben Sie hier Informationen darüber an, wer die Konfiguration der Nachbarstationen pflegt.

Line Type

- **Unterbrochen**: Mit dieser Option können Sie wählen, ob die Verbindung automatisch beendet werden soll (suspended) oder nicht.
- **Autodial**: Wenn diese Option ausgewählt ist, baut die OFTP Middleware automatisch eine Verbindung auf, wenn eine Nachricht zu senden ist.
- **Parallele Session**: Der Standard ist „1“. Mit dem Wert „-1“ wird der Wert aus der globalen Variablen MAXSESSIONS übernommen, der in der OFTP Middleware gesetzt wird.
- **Verzögerung vor dem Wählen**: Geben Sie hier eine ganze Zahl an. Die Zahl gibt die Sekunden an, die zwischen zwei Versuchen vergehen soll, eine Verbindung aufzubauen. Der Standard ist „0“.

Protokolltyp der Nachbarstation

Der weitere Aufbau des Dialogs für die Nachbarstation hängt davon ab, welches Verbindungsprotokoll Sie ausgewählt haben.

24.6.1.1 Nachbarstation über TCP/IP

- **IP Adresse**: Geben Sie entweder den Namen oder die IP Nummer des Rechners an, über den die TCP/IP Verbindung läuft.
- **Port**: Geben Sie hier die Portnummer für die TCP/IP Verbindung an.

24.6.1.2 Nachbarstation über SNA LU6.2

SNA (Systems Network Architecture) ist eine Netzwerkarchitektur, die von IBM entwickelt wurde. Mit der LU Type 6.2 kann der parallele Aufbau von Verbindungen einer einzelnen Logical Unit zu mehreren Anwendungsprogrammen erreicht werden.

- **LU Name**: Name der Logischen Einheit (logical unit = LU)
- **Netzwerk ID**: Die Netzwerk ID (auch NETID) spezifiziert ein SNA Netzwerk. Es ist eine ein- bis achtstellige alphanumerische Zeichenkette.
- **TP Name**: TP steht für Transaction Program. „TP Name“ spezifiziert den Namen eines Advanced Program-to-Program Communication (APPC) Transaction Programms auf einem Datenbankserver. Beispiele sind RVSOFTP oder DB2DRDA.
- **Modus**: Der Name des Modus muss eindeutig sein und den SNA Namenskonventionen entsprechen, beispielsweise „LU62PS“.
- **Benutzer ID/Passwort**: Benutzer ID entspricht einem Login-Namen. Einige SNA Netzwerke erfordern einen Authentifizierung.
- **Sicherheit**: Einige SNA Netzwerke erfordern Authentifizierung. Wenn diese gefordert ist, muss die Checkbox ausgewählt werden. Dadurch werden Benutzer ID und Passwort zu den übertragenen Daten hinzugefügt.
- **Sync Level**: Die hier möglichen Werte sind 0 und 1 oder NONE und CONFIRM. 0 oder NONE bedeutet, dass die Verarbeitung nicht bestätigt werden muss. Bei 1 oder CONFIRM wird die Verarbeitung bestätigt.

24.6.1.3 Nachbarstation über ISDN

Wenn Sie eine Nachbarstation konfigurieren, die über ISDN verbunden ist, wird automatisch das X.25 Protokoll zur Konfiguration hinzugefügt, da ISDN allein nicht zum Aufbau einer OFTP-konformen Verbindung ausreicht.

- **ISDN Adresse**: Hier sind die letzten drei Zahlen der ISDN Adresse gemäß Richtlinie E.164 der ITU-T gemeint, die als Filter benutzt werden. Damit können Sie die Endgeräte anhand ihrer letzten drei Ziffern unterscheiden.

- **IP Adresse**: Geben Sie die IP Adresse oder den Rechnernamen Ihres ISDN Routers an (nur nötig, wenn Sie mehrere ISDN Router verwenden).
- **Typ**: Üblicherweise RCAPI1. Dabei steht RCAPI für „Remote Common ISDN Application Programming Interface“, die Zahl steht für die Nummer des Router Controllers, hier 1.
- **Protokoll**: EDSS1 (Signalisierungsprotokoll „Euro Digital Subscriber Signaling System No. 1")
- **Paketgröße**: Üblicherweise die Paketgröße 128.
- **Facilities**: Meist nicht angegeben.
- **Benutzerdaten**: Meist nicht angegeben.

Wenn Sie „Facilities“ und „Benutzerdaten“ angeben möchten, dann als HEX-Code. Die Daten werden nur für X.25 über ISDN Übertragungen benötigt, wenn Ihr X.25 Provider dies fordert.

- **Terminal-Kennung**: Auch Terminal-ID genannt. Eindeutige Nummer Ihres ISDN-Endgeräts.

24.6.1.4 Nachbarstation über X.25

X.25 ist ein internationaler Telekommunikationsstandard, der von der Internationalen Fernmeldeunion verabschiedet wurde. Weitere Informationen finden Sie unter www.itu.int

- **X.25 Adresse**: Jeder Rechner, der in einem X.25 Netzwerk kommunizieren soll, muss eine X.25 konforme Adresse haben. Eine X.25 Adresse ist eine Folge von Ziffern. Die ersten Ziffern geben den Ländercode an, die darauffolgenden Ziffern bestimmen das Netzwerk innerhalb des Landes und die letzten Ziffern die Rechnernummer.
- **Unteradresse**: Wenn unter einem Anschluss mehrere Rechner existieren (Subnetz) dann bestimmen die letzten drei Ziffern der X.25 Adresse den Rechner im Subnetz.
- **Facilities**: Meist nicht angegeben.
- **Benutzerdaten**: Meist nicht angegeben.

Wenn Sie „Facilities“ und „Benutzerdaten“ angeben möchten, dann als HEX-Code. Die Daten werden nur für X.25 über ISDN Übertragungen benötigt, wenn Ihr X.25 Provider dies fordert.

- **Timeout**: Zeit in Sekunden, nach welcher der Versuch, eine Verbindung aufzubauen, abgebrochen wird.
- **Session**: Maximale Anzahl der parallel aktiven Empfänger. Diesen Wert müssen Sie von Ihrem Geschäftspartner erfragen. Der Standard ist 1.

- **DBit**: (Delivery confirmation bit) Wenn Sie diese Option auswählen, fordert die sendende Station eine Empfangsbestätigung an.
- **Geschlossene Benutzergruppe**: (closed user group, CUG) Ist eine Station Mitglied in einer CUG, dann werden Nachrichten nur zwischen Stationen gesendet und empfangen, die in derselben CUG sind.
- **Virtuelle Schaltung**: Bei X.25 sind virtuelle Schaltungen (virtual circuits) möglich. Dabei wird alles außer der Übertragungsroute festgelegt. Damit ist es möglich, die Route bei Bedarf zu wählen und über mehrere Kanäle durch Multiplexing die Übertragungsrate zu erhöhen. Sie können zwischen den Parametern S (switched virtual circuit, SVC) und P (permanent virtual circuit, PVC) wählen. PVC-Verbindungen sind beispielsweise bei Banken gängig, SVC-Verbindungen werden genutzt, wo eine dauerhafte Verbindung nicht notwendig ist.

Wenn Sie eine Nachbarstation hinzufügen die über X.25 kommuniziert, dann werden keine Optionen für das ISDN Protokoll angezeigt. Für X.25 wird dann die Option „Gerätename“ hinzugefügt.

- **Gerätename**: Benennt das Gerät (device name). Geben Sie eine beliebige alphanumerische Zeichenkette ein. Der hier angegebene Namen muss mit dem Namen übereinstimmen, der in der Konfiguration des Geschäftspartners steht.

24.6.1.5 Virtual - Virtuelle Nachbarstation

Mit einer virtuellen Station kann eine Anwendung oder ein Benutzer konfiguriert werden. Wird eine Nachricht für eine virtuelle Station empfangen, wird diese Nachricht lokal an den Benutzer oder das Programm weitergereicht. Ebenso kann eine virtuelle Station auch Nachrichten senden. Dazu wird die Nachricht vom Benutzer oder Programm an die OFTP Middleware übergeben. Diese versendet die Nachricht und trägt als Absender die virtuelle Station ein. Auch virtuelle Stationen sind eindeutig adressierbar da sie die eindeutige Odette ID haben. Für Geschäftspartner stellen sich virtuelle Stationen wie Stationen dar, die routed sind.

Da nur die Odette ID bei einer virtuellen Station wichtig ist, können die Felder Name, Telefon, Bemerkungen mit Daten gefüllt werden, die für die Administration wichtig sind, wie beispielsweise Name und Telefonnummer des Sachbearbeiters.

24.6.1.6 Routed Station

Wenn Sie eine Nachbarstation markieren, können Sie aus dem Kontextmenü die Option „Routed Station hinzufügen“ auswählen. Nachrichten an eine und von einer „Routed Station“ werden von der Nachbarstation weitergeleitet, unter der sie angeordnet sind. Die konkrete Adresse einer Routed Station ist für Geschäftspartner transparent. Wenn Sie eine Routed Station anlegen, zeigt der Dialog auf der rechten Seite die folgenden Parameter:

- **RvsNeighbourSid**: Der Name wird mit dem Namen der Nachbarstation vorbelegt, unter der die Routed Station angelegt wurde.
- **Name, Telefon, Bemerkungen**: Diese Angaben sind optional. Hier können beispielsweise die Daten eines Sachbearbeiters hinterlegt werden.
- **Odette ID**: Eindeutige Zeichenkette mit max. 25 Zeichen. Das erste Zeichen ist immer ein „O“.
- **Eingangsbestätigung (EERP_in)**: (End to End Response, Empfangsbestätigung) Die EERP_in wird vom Geschäftspartner gesendet. Die Combobox bietet die Einträge NEVER und NORMAL.
- **Ausgangsbestätigung (EERP_out)**: Eine Empfangsbestätigung wird von der lokalen Station an den Geschäftspartner gesendet. Standardmäßig ist die Option IMMEDIATE (siehe unten) vorausgewählt. Die folgenden Möglichkeiten stehen zur Verfügung. NEVER, NORMAL, SYNC, IMMEDIATE, HOLD und HOLD/IMMEDIATE.

→ Siehe *Nachbarstationen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 24.6.1, S. 237)* für Infos über die Einträge, die für die Empfangsbestätigungen gewählt werden können.

24.6.1.7 Verbindung aktivieren

Wenn Sie eine Nachbarstation markieren, können Sie aus dem Kontextmenü die Option „Verbindung aktivieren“ auswählen. Mit dieser Option können Sie testen, ob eine Verbindung mit den angegeben Daten zur konfigurierten Nachbarstation aufgebaut werden kann.

24.6.1.8 Entfernen

Wenn Sie eine Nachbarstation markieren, können Sie aus dem Kontextmenü die Option „Entfernen“ auswählen.

Wenn Sie diese Option wählen, wird die ausgewählte Nachbarstation sofort entfernt. Das Löschen der Nachbarstation wird direkt durchgeführt, ohne dass Sie den „Stationen“ Dialog bestätigen müssen. Diese Veränderung löscht auch die ausgewählte Nachbarstation in der rvs Software.

24.6.2 Empfänger

Dieser Abschnitt erläutert die folgenden Themen:

Ein Empfänger ist eine Verbindungskonfiguration, die auf einem der Protokolle: X.25, lu6.2, TCP/IP oder ISDN beruht. Wenn Sie den „Stationen“ Dialog aufrufen und links den Eintrag „LOC“ auswählen, werden auf der rechten Seite des Dialogs alle Empfänger aufgelistet. Sie können Empfänger hinzufügen und entfernen.

Empfänger entfernen

Wenn Sie diese Option auswählen, können Sie aus der Liste aller konfigurierten Empfänger auswählen, welchen Sie entfernen wollen.

Vergewissern Sie sich, dass über den Empfänger den Sie löschen wollen, keine Nachrichten mehr ausgetauscht werden.

Empfänger hinzufügen

Empfänger werden der lokalen Station hinzugefügt. Wie viele Empfänger Sie hinzufügen können, entnehmen Sie dem Handbuch Ihrer eingesetzten OFTP Middleware.

Protokoll des Empfängers: Zur lokalen Station können mehrere Empfänger hinzugefügt werden, die über verschiedene Protokolle angebunden sind. Sie können jeweils ein oder mehrere Empfänger vom Typ X.25, lu6.2, TCP/IP oder ISDN hinzufügen.

24.6.2.1 X.25 Empfänger konfigurieren

X.25 ist ein internationaler Telekommunikationsstandard, der von der Internationalen Fernmeldeunion verabschiedet wurde.

Weitere Informationen finden Sie unter www.itu.int

- **VektorPosition**: Standard ist „2“.
- **X.25 Adresse**: Jeder Rechner, der in einem X.25 Netzwerk kommunizieren soll, muss eine X.25 konforme Adresse haben. Eine X.25 Adresse ist eine Folge von Ziffern. Die ersten Ziffern geben den Ländercode an, die darauffolgenden Ziffern bestimmen das Netzwerk innerhalb des Landes und die letzten Ziffern die Rechnernummer.
- **Unteradresse**: Wenn unter einem Anschluss mehrere Rechner existieren (Subnetz) dann bestimmen die letzten drei Ziffern der X.25 Adresse den Rechner im Subnetz.
- **Facilities**: Meist nicht angegeben.
- **Benutzerdaten**: Meist nicht angegeben.

Wenn Sie „Facilities“ und „Benutzerdaten“ angeben möchten, dann als HEX-Code. Die Daten werden nur für X.25 über ISDN Übertragungen benötigt, wenn Ihr X.25 Provider dies fordert.

- **Timeout**: Zeit in Sekunden, nach welcher der Versuch, eine Verbindung aufzubauen, abgebrochen wird.
- **Session**: Maximale Anzahl der parallel aktiven Empfänger. Erfragen Sie den Wert von Ihrem Geschäftspartner. Der Standard ist 1.
- **DBit**: (Delivery Confirmation Bit) Wenn ausgewählt, dann fordert die sendende Station eine Empfangsbestätigung an.
- **Geschlossene Benutzergruppe**: (Closed User Group, CUG). Wenn eine Station Mitglied in einer CUG ist, dann werden Nachrichten nur zwischen Stationen gesendet und empfangen, die in derselben CUG sind.
- **Virtuelle Schaltung**: (Virtual circuits). Dabei wird alles außer der Übertragungsroute festgelegt. Damit ist es möglich, die Route bei Bedarf zu wählen und über mehrere Kanäle durch Multiplexing die Übertragungsrate zu erhöhen. Sie können zwischen den Parametern S (switched virtual circuit, SVC) und P (permanent virtual circuit, PVC) wählen. PVC-Verbindungen sind beispielsweise bei Banken gängig, SVC-Verbindungen werden genutzt, wo eine dauerhafte Verbindung nicht notwendig ist.
- **Gerätename**: Benennt das Gerät (device name). Geben Sie eine beliebige alphanumerische Zeichenkette ein. Der hier angegebene Namen muss mit dem Namen übereinstimmen, der in der Konfiguration des Geschäftspartners steht.
- **Alias**: Wenn mehr als ein Router im Netzwerk existiert, geben Sie den Namen oder IP Adresse des Routers an, über den die Nachrichten geleitet werden sollen.
- **Aktiv**: Wird die OFTP Middleware gestartet, ist der Empfänger mit dieser Option nach dem Start sofort empfangsbereit.

24.6.2.2 ISDN Empfänger konfigurieren

- **ISDN Adresse**: Hier sind die letzten drei Zahlen der ISDN Adresse gemäß Richtlinie E.164 der ITU-T gemeint, die als Filter benutzt werden. Damit können Sie die Endgeräte anhand ihrer letzten drei Ziffern unterscheiden.
- **IP Adresse**: Geben Sie die IP Adresse oder den Rechnernamen Ihres ISDN Routers an.
- **Typ**: Üblicherweise RCAPI1 (Remote Common ISDN Application Programming Interface). Die Zahl steht für die Nummer des Router Controllers, hier die 1.
- **Protokoll**: Geben Sie hier EDSS1 an, dies steht für das Signalisierungsprotokoll „Euro Digital Subscriber Signaling System No. 1“
- **Paketgröße**: Hier wird üblicherweise die Paketgröße **128** angegeben.
- **Facilities**: Meist nicht angegeben.
- **Benutzerdaten**: Meist nicht angegeben.

Wenn Sie „Facilities“ und „Benutzerdaten“ angeben möchten, dann als HEX-Code. Die Daten werden nur für X.25 über ISDN Übertragungen benötigt, wenn Ihr X.25 Provider dies fordert.

- **Terminal-Kennung**: Auch Terminal-ID) Eindeutige Nummer Ihres ISDN-Endgeräts
- **Timeout**: Zeit in Sekunden, nach welcher der Versuch, eine Verbindung aufzubauen, abgebrochen wird.
- **RCV Timeout**: RCV steht für „receive“. Zeit in Sekunden, für die Aufrechterhaltung des Empfang. „0“ bedeutet, es existiert kein Timeout.
- **Aktiv**: Die unter „LOC“ konfigurierten Empfänger werden mit dieser Checkbox aktiviert.

24.6.2.3 TCP/IP Empfänger konfigurieren

- **IP Adresse**: Geben Sie die IP Adresse oder den Rechnernamen Ihres ISDN Routers an.
- **Port**: Geben Sie die Portnummer an, über die Nachrichten empfangen werden.
- **Max. Eingangssessions**: Maximale Anzahl von parallelen Eingangssessions; mit „-1“ wird auf den globalen Wert des Parameters MAXSESSIONS zugegriffen.
- **Max. Ausgangssessions**: Maximale Anzahl von parallelen Ausgangssessions; mit „-1“ wird auf den globalen Wert des Parameters MAXSESSIONS zugegriffen.

- **Aktiv**: Die unter „LOC“ konfigurierten Empfänger werden mit dieser Checkbox aktiviert.

24.6.2.4 SNA LU6.2 Empfänger konfigurieren

Mit der Logical Unit LU Type 6.2 kann der parallele Aufbau von Verbindungen einer einzelnen Logical Unit zu mehreren Anwendungsprogrammen erreicht werden.

- **VectorPosition**: Ein Datensegment in einer SNA-Message wird auch als Vektor bezeichnet. Ein SNA-Vektor besteht aus einem Längenfeld, einem Typfeld und vektorspezifischen Daten. Der Standardwert ist „2“.
- **LU Name**: Name der Logischen Einheit (logical unit = LU)
- **Netzwerk ID**: Die Netzwerk ID (auch NETID) spezifiziert ein SNA Netzwerk. Es ist eine ein- bis achtstellige alphanumerische Zeichenkette.
- **TP Name**: TP steht für Transaction Program. TP Name spezifiziert den Namen eines Advanced Program-to-Program Communication (APPC) Transaction Programms auf einem Datenbankserver. Beispiele sind RVSOFTP oder DB2DRDA.
- **Modus**: Der Name des Modus muss eindeutig sein und den SNA Namenskonventionen entsprechen, beispielsweise „LU62PS“.
- **Benutzer ID/Passwort**: Benutzer ID entspricht einem Login-Namen. Einige SNA Netzwerke erfordern einen Authentifizierung.
- **Sicherheit**: Einige SNA Netzwerke erfordern Authentifizierung. Wenn diese gefordert ist, muss die Checkbox ausgewählt werden. Dadurch werden Benutzer ID und Passwort zu den übertragenen Daten hinzugefügt.
- **Sync Level**: Mögliche Werte:
  - 0 oder NONE: Verarbeitung muss nicht bestätigt werden
  - 1 oder CONFIRM: Verarbeitung wird bestätigt

25 OFTP2 Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwenden Sie den OFTP2 Connector nicht gemeinsam mit dem OFTP Connector!

Verwendung

Der OFTP2 Connector wird genutzt, um Geschäftsdaten gemäß dem Odette File Transfer Protocol Version 2.0 (OFTP2) auszutauschen.

OFTP2 bietet die Möglichkeit, eine Verbindung nach dem Abbruch wieder aufzusetzen, ohne die gesamte Datei erneut zu übertragen. Es können ASCII- und EBCDIC-Dateien übertragen werden.

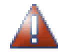
→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

25.1 Voraussetzungen

- rvsEVO der Firma T-Systems International GmbH muss installiert und konfiguriert sein.

  Dabei müssen Übertragungsparameter eingestellt und Stationen mit einer definierten Station-ID eingerichtet werden. Die Software besteht aus dem rvsEVO Server und einer Client-Komponente. Sie kann sowohl über das Benutzerinterface der Client-Komponente als auch über das Benutzerinterface des rvsEVO Servers konfiguriert werden.

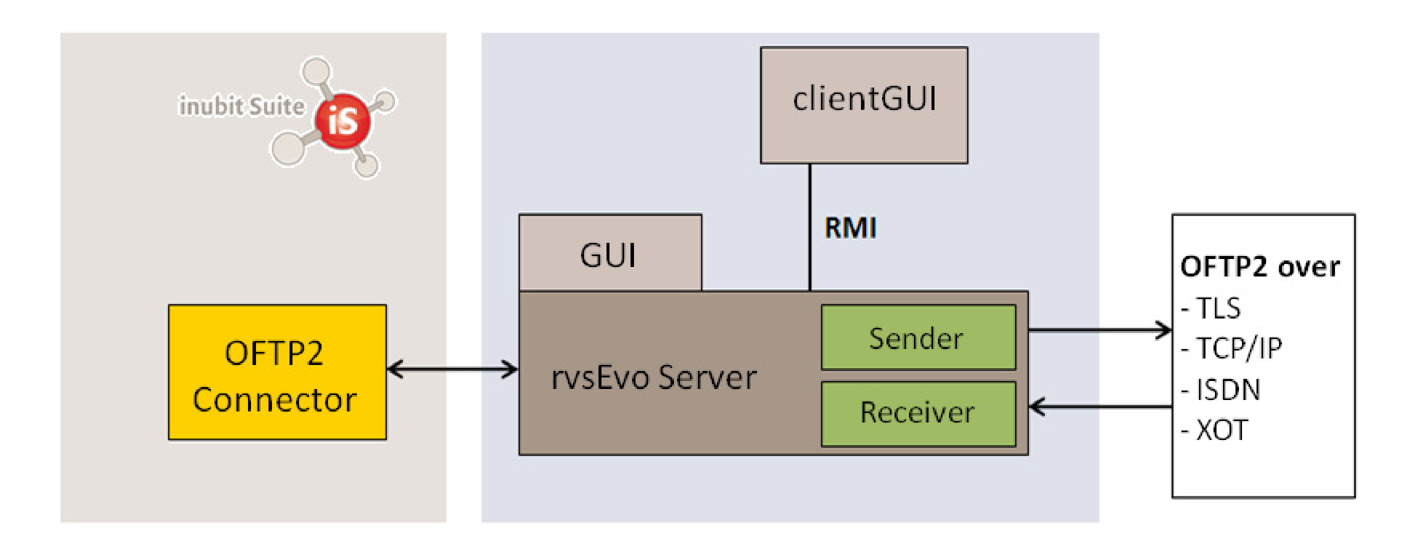


Erfragen Sie unbedingt bei inubit AG, welche rvs-Version Sie für Ihre Installation benötigen.

- Sie haben die folgenden Dateien der rvs-Client-Komponente in das Verzeichnis `<iS-installdir>/server/Tomcat/webapps/ibis/WEB-INF/lib` der inubit Process Engine kopiert:
  - `rvs-identity-data-1.0.jar`
  - `rvs-util-1.3.1.jar`
  - `rvs-client-api-1.3.1.jar`
  - `rvs-common-exception-1.0.jar`
  - `rvs-data-1.3.1.jar`
  - `rvs-datahelper-1.5.jar`
  - `rvs-evo-server.jar`

Für weitere Informationen beachten Sie die Dokumentation der rvs Software.

25.2 Funktionsprinzip



Nachrichtenfluss

- **Nachricht versenden mit Output Connector**
  Der OFTP2 Connector übergibt die Nachrichten noch auf der inubit Process Engine an rvsEVO. Der Versand kann mit Hilfe des Schedulers so konfiguriert werden, dass die Nachricht sofort verschickt wird, wenn sie im Workflow am OFTP2 Connector ankommt, oder nur zu einer vereinbarten Zeit oder in einem Zeitintervall.

rvsEVO verschickt die Nachricht über das vereinbarte Protokoll an die vom Empfänger eingerichtete Station.

- **Nachricht empfangen**
  - **Input Connector**
    Der OFTP2 Connector prüft aktiv in einem festgelegten Zeitintervall, ob Nachrichten vorhanden sind.
  - **Input Listener Connector**
    Der OFTP2 Connector wartet auf der konfigurierten Verbindung auf das Eintreffen von Nachrichten.

Abhängig davon, wie Ihr Konnektor konfiguriert ist, findet die Kommunikation synchron oder asynchron statt, siehe *Datenübertragungsmodi (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 25.3, S. 251)*.

25.3 Datenübertragungsmodi

Der OFTP2 Connector unterstützt folgende Datenübertragungsmodi:

- **Asynchrone Übertragung mit Input Connector**
  Der Input Connector holt die Daten entsprechend der Konfiguration im *Dialog „Zeitgesteuerte Verarbeitung“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1.1.3, S. 22)*“.
  rvsEVO sendet eine Bestätigung (EERP), wenn die Verbindung weiter besteht oder beim nächsten Verbindungsaufbau. Dies entspricht der Option „Normal“ für den Parameter EERP_out.
  → Siehe Parameter EERP_out im Abschnitt *Nachbarstationen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 25.6.2, S. 258)*.

Die Bestätigung garantiert dem Absender der Nachricht nicht, dass die Nachricht auf der inubit Process Engine angekommen ist! Achten Sie darauf, dass die zulässige Größe der Datenbank im Verlauf des Betriebs nicht überschritten wird.

Wenn Sie mehr Informationen als den Dateinamen erhalten möchten, dann müssen Sie die rvsEVO-Konfiguration entsprechend ändern.

- **Synchrone Übertragung durch rvsEVO mit Input Listener Connector**
  rvsEVO bestätigt per EERP-out den Empfang der Nachricht.

Dieses Verfahren ist in Bezug auf den OFTP2 Connector halbsynchron, weil die Bestätigung versendet wird, bevor der OFTP2 Connector bestätigen kann, dass die Nachricht auf der inubit Process Engine vorliegt.

- **Synchrone Übertragung mit Input Listener Connector**

  Die Empfangsbestätigung wird verschickt, wenn sie vollständig auf der inubit Process Engine eingetroffen ist.

  In der Konfiguration der Nachbarstation muss der Parameter EERP_out auf „HOLD“ gesetzt werden.

  → Siehe Parameter EERP_out im Abschnitt *Nachbarstationen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 25.6.2, S. 258)*.

  Der Job (Nachrichtenempfang) bleibt aktiv, bis die Empfangsbestätigung (EERP_out) gesendet wurde.

- **Synchrone Übertragung mit Output Connector**

  Setzen Sie beim Konfigurieren des Konnektors ein Timeout, damit im Fehlerfall keine Blockade durch das Warten entsteht. Berücksichtigen Sie bei dem Wert die Nachrichtengröße und den Übertragungsweg.

  Kontrollieren Sie die Konfiguration der Gegenstelle: wenn dort die EERP-Nachricht auf „hold“ steht, kann es auch zu einer Blockade kommen.

  Send Job bei erfolgreichem/fehlgeschlagenem Senden: `rvs_is_send`.

25.4 rvsEVO Monitoring

Mit dem Kommandozeilentool „StartCLI“ können Sie u. a. die Verbindung der inubit Process Engine mit dem rvsEVO Server prüfen.

Optionen

Option	Information
`-r,--rvs`	Verbindung der inubit Process Engine mit dem rvsEVO Server prüfen
`--rhost`	rvsEVO Host
`--rlogin`	rvsEVO Login
`--rmwname`	Name des rvsEVO Servers
`--rpassword`	rvsEVO Passwort

→ Siehe *Remote-Überwachung mit Command Line Interface (StartCLI) (Process Engine: Administrator- und Entwickler-Guide, Kap. 7.10, S. 108).*

25.5 Dialog „OFTP Datenaustausch-Konfiguration“

Dieser Dialog unterscheidet sich für Input Connector, Input Listener Connector und Output Connector.

Grundeinstellungen

- **rvsEVO-Serveradresse**
  Geben Sie hier die Rechnernamen oder die IP-Nummer an, z. B. „MyServer“ oder „123.456.7.8“. Bei der Eingabe müssen Sie die Anführungsstriche weg lassen. Wenn die RMI Registry nicht auf Port 3755 läuft, geben Sie die Portnummer an, die mit einem Doppelpunkt vom Rechnernamen getrennt wird, z. B. „MyServer:1234“.
- **rvsEVO-Name**
  Hier wird üblicherweise die Zeichenkette „rvsmw“ benutzt. Wenn Sie einen anderen Namen für rvsEVO vergeben haben, übernehmen Sie diesen hier.
- **Anmeldung**
  Geben Sie hier die Benutzernamen an, mit dem Sie sich bei rvsEVO anmelden. Groß- und Kleinschreibung wird unterschieden.
- **Passwort**
  Geben Sie hier Ihr Passwort ein.
- **Dateiname**
  Der Name wird für die OFTP2 Übertragung benutzt und muss dem Empfänger bekannt gemacht werden, damit dieser auf die Datei zugreifen kann.

Im Dateinamen können Sie Wildcards oder reguläre Ausdrücke verwenden:

- **Wildcards**
  - Fragezeichen (?): Steht für genau ein beliebiges Zeichen.
  - Asterisk (*): Steht für Null bis n beliebige Zeichen.

  Die Wildcards können in beliebiger Kombination und Häufigkeit benutzt werden.

  Wenn Wildcards angegeben sind, dann holt z. B. ein Input Connector nur Dateien, die dem angegebenen Wert entsprechen, alle anderen Dateien werden ignoriert. Um alle Dateien zu empfangen, geben Sie * ein.
- **Reguläre Ausdrücke**

Reguläre Ausdrücke müssen in Schrägstriche eingeschlossen werden, z. B. `/(\d\d).(\d\d).(\d\d\d\d)/`.

Für Informationen über reguläre Ausdrücke siehe z. B. http://www.perl.com/doc/manual/html/pod/perlre.html oder http://www.regular-expressions.info/tutorial.html.

Windows: Das Windows Betriebssystem ermöglicht es, Dateien auszublenden. Wenn ausgeblendete Dateien einem Wildcard-Kriterium genügen, werden sie auch übertragen. Stellen Sie sicher, dass die Dateien, die Sie im Verzeichnis des Explorers sehen können, auch alle Dateien sind, die hier enthalten sind. Blenden Sie dazu alle Dateien ein (im Explorer Optionen „Extras“ - „Ordneroptionen“).

- **Inbox-Unterverzeichnis** (nicht beim Output Connector)

  Geben Sie den Verzeichnisnamen an, den Sie in rvsEVO unter „Inbox“ angelegt haben und in das die eingehenden Dateien abgelegt werden sollen.

  Der Ordner `Inbox` ist identisch mit dem Ordner `<rvsEVO-installdir>/files/inbox` auf dem rvsEVO Server. Wenn Sie kein anderes Verzeichnis konfiguriert haben, werden hier die eingehenden Nachrichten abgelegt.
- **Virtueller Dateiname** (nicht beim Input Connector)

  (Virtual dataset name, VDSN) Unter diesem Namen wird die Datei versendet. Wird kein Wert angegeben, so wird der VDSN-Name aus dem Originaldateinamen gebildet.
- **Dateibeschreibung** (nicht beim Input Connector)

  Nähere Erläuterung des virtuellen Dateinamens
- **Label** (nicht beim Input Connector)

  Zur Kennzeichnung von Dateien, die in der gleichen Gruppe versendet werden sollen.
- **Externe JobID** (nicht beim Input Connector)

Externe Kennung des Jobs.

- **Konvertierungstabelle**
  Zur Auswahl von Datenformaten (links: Quellformat, rechts: Zielformat).
- **Empfängerstations-ID** (nicht beim Input Connector)
  Eine Stations-ID (SID) einer Nachbarstation kann bis zu 16 Zeichen umfassen. Diese ID wird nur lokal genutzt. Die ID ist frei wählbar und kann aus Zahlen und Buchstaben bestehen. Diese SID muss zuerst in der rvsEVO Software bzw. über den Dialog „Stationen“ angelegt werden und kann dann hier übernommen werden. Wenn Sie die Nachbarstation über den Dialog „Stationen“ auswählen, wird deren ID automatisch übernommen.
- **Stationen**
  Der Dialog „Stationen“ kann nur aufgerufen werden, wenn bereits gültige Verbindungsdaten zu einer rvsEVO-Installation im Dialogabschnitt „Grundeinstellungen“ eingegeben wurden.
- **Ohne Kopie**
  Beim Erzeugen eines Sendeeintrags wird standardmäßig die zu versendende Datei ins Outbox-Verzeichnis von rvsEVO kopiert und von hier versendet. Mit der Option „Ohne Kopie“ kann dies unterbunden werden. In diesem Falle wird die Originaldatei nach erfolgreichem Versand gelöscht.

Diese Option hat keine Auswirkung, wenn die Datei im Format U (unstrukturiert) mit Kodekonvertierung versendet wird.

- **Datei auf dem Server löschen**
  Nach der Übertragung wird die Datei auf dem rvsEVO Server gelöscht, wenn diese Option ausgewählt ist.
- **Serialisierung**
  Aktiviert/deaktiviert das exakte Einhalten der Reihenfolge der zu sendenden Dateien. Nachrichten, die zusammen versendet werden sollen, müssen dasselbe Label besitzen.
- **Synchroner Modus**
  Wenn die Checkbox markiert ist, wird der synchrone Datenübertragungsmodus aktiviert.

Format

(nur Output Connector)

Formatinformation für die Datei an, die übertragen werden soll:

- **Format**
  - **Text**: Zum Übertragen von Textdateien im ASCII-Format
  - **Feste Sätze**: Zum Übertragen von Dateien mit festen Satzlängen.
  - **Variable Sätze**: Zum Übertragen von Dateien mit variablen Satzlängen.

- **Unstrukturiert**: Zum Übertragen binärer Dateien.
- **Maximale Satzlänge**
  Für feste und variable Sätze können Sie eine maximale Satzlänge angeben.

Sicherheit

(nur bei Output Connector)

- **Sicherheitsmerkmale**
  Wählen Sie die gewünschte Sicherheitsstufe.
- **Kompression**
  Checkbox zum Aktivieren/Deaktivieren der Datenkompression.
- **Verschlüsselung**
  Checkbox zum Aktivieren/Deaktivieren der Dateiverschlüsselung. Hierbei handelt es sich nicht um die Session-Verschlüsselung von TLS.
  → Weitere Informationen zur Verschlüsselung entnehmen Sie der rvsEVO-Dokumentation.
- **Verschlüsselungsalgorithmus**
  Zur Auswahl des Verschlüsselungsalgorithmus (Standard: DES_EDE3_CBC).
- **Dateisignatur**
  Checkbox zum Aktivieren/Deaktivieren der Verwendung einer Dateisignatur.
- **Signierten EERP/NERP beantragen**
  Checkbox zum Beantragen/Nichtbeantragen eines signierten EERP/NERP.

25.6 Dialog „Stationen“

Dieser Abschnitt erläutert die folgenden Themen:

Wenn Sie den „Stationen“ Dialog aufrufen, verbindet sich der OFTP2 Connector über die von Ihnen angegebenen „Grundeinstellungen“ mit der rvsEVO-Installation und erhält dort die Konfigurationsdaten, die im „Stationen“ Dialog angezeigt werden.

Beachten Sie, dass Einstellungen im Dialog „Stationen“ die Konfiguration in der rvsEVO-Installation ändern, wenn Sie den Dialog mit „OK“ beenden.

25.6.1 Lokale Station

LOC

„LOC“ ist die lokale Stationsdatei. Wenn Sie diesen Eintrag auswählen, können Sie im rechten Fenster die Konfiguration der lokalen Station angeben. Die Konfigurationsdaten müssen mit denen übereinstimmen, die Sie in Ihrer rvsEVO-Administrator-Konsole angegeben haben.

Die lokale Station muss immer den Namen „LOC“ (in Großbuchstaben) haben.

Wenn Sie „LOC“ markiert haben, steht Ihnen ein Kontextmenü mit folgenden Befehlen zur Verfügung:

- Nachbarstationen hinzufügen
- Empfänger hinzufügen
- Empfänger entfernen
- Verbindung aktivieren

Allgemeine Stationsparameter

Der Bereich „Lokale Station“ bzw. „Nachbarstation“ enthält neben der RvsSid (Pflicht) Angaben zur Station, wie Name, Telefon, Unternehmen, Ort, Straße, Kontaktperson, E-Mail.

Bereich OFTP

Im Bereich OFTP konfigurieren Sie Kommunikationsparameter für die lokale Station und alle Nachbarstationen.

Für die lokale Station konfigurieren Sie lediglich die drei Parameter Odette ID, PKI und Zertifikatsvalidierung.

→ Die Beschreibung der Parameter entnehmen Sie der rvsEVO-Dokumentation.

- **Odette ID**: Dies ist eine eindeutige Zeichenkette, die aus maximal 25 Zeichen besteht. Das erste Zeichen ist immer ein „O“.

  Eine Odette ID erhalten Sie unter www.vda.de

- **CertificationValidationType**

  Zum Festlegen der Art der Zertifikatsvalidierung:

  - CERT_PATH (schwache Validierung): Validierung über ROOT-Zertifikate und/oder die Zertifikate der CAs

  - CRL (stärkere Validierung): Validierung über die Certificate Revocation List
  - OCSP (stärkste Validierung): Validierung über das Online Certificate Status Protocol
  - NONE: keine Validierung
- **UsePki**
  - Aktivierte Checkbox: Aktiviert vorhandene PKI mit LDAP
  - Deaktivierte Checkbox: Zertifikate aus der lokalen Schlüsselverwaltungsdatei verwenden

Empfängerbereiche

Die Konfiguration ist abhängig vom gewählten Protokoll.

→ Siehe *Empfänger (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 25.6.3, S. 264).*

25.6.2 Nachbarstationen

Dieser Abschnitt erläutert die folgenden Themen:

Eine Nachbarstation ist die Station eines Geschäftspartners, mit dem Sie Nachrichten austauschen wollen.

Nachbarstation hinzufügen

Mit dieser Option fügen Sie eine neue Nachbarstation hinzu. Diese wird in einem Verzeichnisbaum unter „LOC“ hinzugefügt. Sie müssen die Konfiguration der Nachbarstation kennen. Wenn Sie die neu hinzugefügte Station auswählen, können Sie die Parameter auf der rechten Seite des Dialogs hinzufügen. Die Stationen können für verschiedene Übertragungsprotokolle konfiguriert werden. Wählen Sie eines der Protokolle: TCP/IP, TLS-, ISDN, XOT-, Proxy TCP/IP- oder Proxy TLS.

Bereich „OFTP“

- **Odette ID** (Pflicht): Eindeutige Zeichenkette, die aus maximal 25 Zeichen besteht. Das erste Zeichen ist immer ein „O“.

- **Passwort empfangen**: Das vom Geschäftspartner erwartete Passwort.
- **Passwort senden**: Das an den Geschäftspartner gesendete Passwort.
- **Größe des Austauschpuffers**: Maximale Größe des Übertragungspuffers in Byte. Mögliche Werte: 0 - 99999.
- **Anzahl der Austauschpuffer**: Maximale Zahl der gesendeten Blöcke ohne Erwartung einer Quittung. Mögliche Werte: 0 - 999.
- **Eingangsbestätigung (EERP_in)**:

  EERP (End to End Response) bedeutet Empfangsbestätigung. Die EERP_in wird vom Geschäftspartner gesendet.
  - **NEVER**: Der Geschäftspartner sendet keine Eingangsbestätigung. Die Übertragung wird mit der korrekten Übermittlung der Nachricht beendet.
  - **NORMAL**: Es wird eine Eingangsbestätigung erwartet. Der Eingang der Bestätigung gehört zur Sendeaufforderung. Eine Sendeaufforderung besteht so lange wie entweder noch Dateien zur Übertragung anstehen oder auf den Eingang von Eingangsbestätigungen gewartet wird. Mit dem Eingang der Bestätigung wird auch die Sendeaufforderung beendet. Die Auswahl NORMAL ist der Standard.
- **Ausgangsbestätigung (EERP_out)**:

  Eine Empfangsbestätigung wird von der lokalen Station an den Geschäftspartner gesendet. Standardmäßig ist die Option IMMEDIATE vorausgewählt:
  - **NEVER**: Es wird keine Empfangsbestätigung gesendet.
  - **NORMAL**: Beim Empfang einer Nachricht wird eine Empfangsbestätigung gesendet. Wenn die Verbindung noch besteht, wird die Bestätigung sofort gesendet. Wenn die Verbindung bereits abgebaut wurde, wird die Bestätigung beim nächsten Verbindungsaufbau gesendet.
  - **SYNC**: Die Empfangsbestätigung wird über dieselbe Verbindung gesendet, über welche die Nachricht eingegangen ist. Das Bestehen der Verbindung wird dabei erzwungen. Es wird geprüft, ob die Empfangsbestätigung erfolgreich versendet wurde, erst dann kann die Verbindung geschlossen werden.
  - **HOLD**: Die Empfangsbestätigung wird erst gesendet, wenn sie explizit von einem Operator in der rvsEVO-Installation freigegeben wurde. Besteht die Verbindung noch, so wird die Empfangsbestätigung sofort gesendet. Wenn die aktuelle Verbindung bereits abgebaut wurde, wird die Empfangsbestätigung gesendet wenn das nächste Mal eine Verbindung erstellt wurde.

> Der OFTP2 Connector erteilt die Freigabe für die rvsEVO-Installation selbständig, es ist keine Interaktion eines Operators erforderlich.

- **RoutingSync**: Die OFTP2-Session zwischen dem Router und dem Sender wird so lange aufrecht erhalten, bis die Empfangsbestätigung vom Empfänger angekommen ist, um diese dann in derselben OFTP2-Session an den Sender zurückzusenden.

Neighbour Station

- Name, Telefon, Bemerkungen und Netzwerk sind optionale Felder. Geben Sie hier an, wer die Konfiguration der Nachbarstationen pflegt.

Line Type

- **Autodial**: Wenn diese Option ausgewählt ist, baut rvsEVO automatisch eine Verbindung auf, wenn eine Nachricht zu senden ist.
- **Parallele Session**: Der Standard ist „1“.

Sicherheit

- **OFTP Version**
  Dieser Parameter muss nur gesetzt werden, wenn der Partner nicht in der Lage ist, die Protokollversion auszuhandeln.
- **SFIDDESC als Dateiname**
  (Start File ID-Description) Wird anstatt VDSN als Dateiname für die Übertragung verwendet.
- **PKI**
  Aktiviert/Deaktiviert die PKI-Anbindung.
- **Zertifikatsvalidierung**
  Legt fest, wie die Zertifikate validiert werden sollen.

Protokolltyp der Nachbarstation

Der weitere Aufbau des Dialogs für die Nachbarstation hängt davon ab, welches Verbindungsprotokoll Sie ausgewählt haben.

25.6.2.1 Nachbarstation über TCP/IP

- **IP Adresse**: Name oder IP-Nummer des Rechners, über den die TCP/IP Verbindung läuft.
- **Port**: Portnummer für die TCP/IP Verbindung.

25.6.2.2 Nachbarstation über ISDN

Wenn Sie eine Nachbarstation konfigurieren, die über ISDN verbunden ist, wird automatisch das X.25 Protokoll zur Konfiguration hinzugefügt, da ISDN allein nicht zum Aufbau einer OFTP2-konformen Verbindung ausreicht.

- **Nummer des zugeordneten Empfängers**: Wird automatisch vergeben.
- **Typ (Pflicht)**: Nicht konfigurierbar.
- **Karten-Nummer (Pflicht)**: Wird automatisch vergeben.
- **ISDN Adresse**: Entspricht der ISDN-Nummer des Partners.
- **Typ**: Üblicherweise CAPI2A.
- **ISDN Protokoll**: Gibt den genutzten ISDN-Standard an.
  - 1TR6: Deutscher Landesstandard
  - E-DSS1: EURO-ISDN (Standard)
- **Paketgröße**: Üblicherweise die Paketgröße 128.
- **ISDN Facilities**: Meist nicht angegeben.
- **ISDN Benutzerdaten**: Meist nicht angegeben.

Wenn Sie „Facilities“ und „Benutzerdaten“ angeben möchten, dann als HEX-Code. Die Daten werden nur für X.25 über ISDN Übertragungen benötigt, wenn Ihr X.25 Provider dies fordert.

→ Weitere Informationen können Sie der rvsEVO-Dokumentation entnehmen.

- **Terminal-Kennung**: Auch Terminal-ID genannt. Eindeutige Nummer Ihres ISDN-Endgeräts.
- **Verbindungs-Timeout**:
  Zeit in Sekunden, nach welcher der Versuch, eine Verbindung aufzubauen, abgebrochen wird.
- **X.25 Adresse**
  15-stellige X.25-DTE-Adresse der Partnerstation. Angabe ist optional. Es wird jedoch empfohlen, hier die ISDN-Nummer einzutragen, da manche Gegenstellen eine X.25-Adresse erwarten.
- **X.25 Paketgröße**
  Größe der Datenpakete bei der Datenübertragung.
  Standard: 128
- **X.25 Facilities**
  Spezielle Informationen oder Anlagen bei X.25-Übertragung
- **X.25 Benutzerdaten**
  Benutzerangaben für X.25-Übertragung
- **X.25 Fenstergröße**

Die Fenstergröße bei X.25/ISDN-Kommunikation ist die Anzahl der Pakete, die ohne Bestätigung verschickt werden können. Die Fenstergröße kann während des Verbindungsaufbaus ausgehandelt werden. Wir empfehlen aber von vornherein die zum Partnernetzwerk passende Fenstergröße zu konfigurieren, z. B. den Wert „7“ für ISDN.

- **X.25 DBit**
  Das D-Bit ist ein Feld im Datenpaket von X.25, das für die Ende-zu-Ende-Bestätigung benutzt wird. Die DTE zeigt damit an, ob sie den Empfang einer Ende-zu-Ende-Bestätigung wünscht.
  Standard: deaktiviert.
- **X.25 Geschlossene Benutzergruppe**:
  ISDN und X.25 bieten die Möglichkeit, eine geschlossene Benutzergruppe zu bilden. Alle Teilnehmer, die zu einer solchen Gruppe gehören, können untereinander über das öffentliche Telekommunikationsnetz kommunizieren. Verbindungsanforderungen an Gruppenmitglieder von Teilnehmern, die nicht Mitglied der geschlossenen Benutzergruppe sind, werden von der Vermittlungsstelle abgewiesen. Standard: deaktiviert.

25.6.2.3 Nachbarstation über XOT

XOT steht für X.25 over TCP. X.25 ist ein internationaler Telekommunikationsstandard, der von der Internationalen Fernmeldeunion verabschiedet wurde.

→ Siehe *XOT Empfänger konfigurieren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 25.6.3.3, S. 266).*

Weitere Informationen finden Sie unter http://www.itu.int.

- **X.25 Adresse**: Jeder Rechner, der in einem X.25 Netzwerk kommunizieren soll, muss eine X.25 konforme Adresse haben. Eine X.25 Adresse ist eine Folge von Ziffern. Die ersten Ziffern geben den Ländercode an, die darauffolgenden Ziffern bestimmen das Netzwerk innerhalb des Landes und die letzten Ziffern die Rechnernummer.
- **X.25 Facilities**: Meist nicht angegeben.
- **X.25 Benutzerdaten**: Meist nicht angegeben.

Wenn Sie „Facilities“ und „Benutzerdaten“ angeben möchten, dann als HEX-Code. Die Daten werden nur für X.25 über ISDN Übertragungen benötigt, wenn Ihr X.25 Provider dies fordert.

- **Timeout**: Zeit in Sekunden, nach welcher der Versuch, eine Verbindung aufzubauen, abgebrochen wird.

- **DBit**: (Delivery confirmation bit) Wenn Sie diese Option auswählen, fordert die sendende Station eine Empfangsbestätigung an.
- **X.25 Paketgröße**

  Größe der Datenpakete bei der Datenübertragung. Standard: 128
- **X.25 Fenstergröße**

  Die Fenstergröße bei X.25/ISDN-Kommunikation ist die Anzahl der Pakete, die ohne Bestätigung verschickt werden können. Die Fenstergröße kann während des Verbindungsaufbaus ausgehandelt werden. Wir empfehlen aber von vornherein die zum Partnernetzwerk passende Fenstergröße zu konfigurieren, z. B. den Wert „7“ (Standard) für ISDN.
- **X.25 DBit**

  Das D-Bit ist ein Feld im Datenpaket von X.25, das für die Ende-zu-Ende-Bestätigung verwendet wird. Die DTE zeigt damit an, ob sie den Empfang einer Ende-zu-Ende-Bestätigung wünscht.

  Standard: deaktiviert.
- **X.25 Modulo**

  In der X.25-Datenübertragung wird ein auf Modulo basierendes Verfahren für die folgerichtige Übertragung von Nutzerinformationen mit Hilfe von Zählern und einem Fenstermechanismus auf der Sicherungsschicht durchgeführt.

  Für X.25 sind zwei Modulo-Varianten bekannt: der Modulo-8-Zähler und der Modulo-128-Zähler. Standard: 8.

25.6.2.4 Routed Station

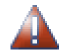
Diese Stationen können Sie derzeit nicht im OFTP2 Connector konfigurieren. Sie können Routed Stations aber direkt in rvsEVO konfigurieren und danach im OFTP2 Connector als Zielstation auswählen.

25.6.2.5 Verbindung aktivieren

Wenn Sie eine Nachbarstation markieren, können Sie aus dem Kontextmenü die Option „Verbindung aktivieren“ auswählen. Mit dieser Option können Sie testen, ob eine Verbindung mit den angegeben Daten zur konfigurierten Nachbarstation aufgebaut werden kann.

25.6.2.6 Entfernen

Wenn Sie eine Nachbarstation markieren, können Sie aus dem Kontextmenü die Option „Entfernen“ auswählen.



Wenn Sie diese Option wählen, wird die ausgewählte Nachbarstation nach der Bestätigung in einem Meldungsfenster gelöscht. Diese Veränderung löscht auch die ausgewählte Nachbarstation in der rvsEVO Software.

25.6.3 Empfänger

Dieser Abschnitt erläutert die folgenden Themen:

Ein Empfänger ist eine Verbindungskonfiguration, die auf einem der folgenden Protokolle beruht: TCP/IP, ISDN, XOT oder TLS. Wenn Sie den „Stationen“ Dialog aufrufen und links den Eintrag „LOC“ auswählen, werden auf der rechten Seite des Dialogs alle Empfänger aufgelistet. Sie können Empfänger hinzufügen und entfernen.

Empfänger entfernen

Wenn Sie diese Option auswählen, können Sie aus der Liste aller konfigurierten Empfänger auswählen, welchen Sie entfernen wollen.

Stellen Sie sicher, dass über den Empfänger den Sie löschen wollen, keine Nachrichten mehr ausgetauscht werden.

Empfänger hinzufügen

Empfänger werden der lokalen Station hinzugefügt. Wie viele Empfänger Sie hinzufügen können, entnehmen Sie dem Handbuch Ihrer eingesetzten OFTP2 Middleware.

Protokoll des Empfängers: Zur lokalen Station können mehrere Empfänger hinzugefügt werden, die über verschiedene Protokolle angebunden sind. Sie können jeweils ein oder mehrere Empfänger vom Typ TCP/IP, ISDN, XOT oder TLS hinzufügen.

25.6.3.1 TCP/IP Empfänger konfigurieren

- **VectorPosition**: Nicht editierbar.
- **Empfänger-Nummer**: Nicht editierbar.
- **IP Adresse**: IP-Adresse oder Rechnernamen Ihres ISDN-Routers.
- **Port**: Portnummer, über die Nachrichten empfangen werden.
- **Max. Eingangssessions**: Maximale Anzahl von parallelen Eingangssessions.
- **Aktiv**: Die unter „LOC“ konfigurierten Empfänger werden mit dieser Checkbox aktiviert.
- **Timeout**: Zeit in Sekunden, nach welcher der Versuch, eine Verbindung aufzubauen, abgebrochen wird.

25.6.3.2 ISDN Empfänger konfigurieren

- **VectorPosition**: Nicht editierbar.
- **Typ**: Üblicherweise CAPI2A.
- **ISDN Protokoll**: Gibt den genutzten ISDN-Standard an.
  - 1TR6: Deutscher Landesstandard
  - E-DSS1: EURO-ISDN (Standard)
- **ISDN Facilities**: Meist nicht angegeben.
- **ISDN Benutzerdaten**: Meist nicht angegeben.
- **X.25 Paketgröße**: Hier wird üblicherweise die Paketgröße 128 angegeben.

Wenn Sie „Facilities“ und „Benutzerdaten“ angeben möchten, dann als HEX-Code. Die Daten werden nur für X.25 über ISDN Übertragungen benötigt, wenn Ihr X.25 Provider dies fordert.

- **ISDN Terminal-Kennung**: Eindeutige Nummer Ihres ISDN-Endgeräts
- **Empfänger-Nummer**: Nicht editierbar.
- **Aktiv**: Die unter „LOC“ konfigurierten Empfänger werden mit dieser Checkbox aktiviert.
- **ISDN Nummer**: Eigene ISDN-Nummer
- **Verbindungs-Timeout**: Zeit in Sekunden, nach welcher der Versuch, eine Verbindung aufzubauen, abgebrochen wird.
- **RCV Timeout**: RCV steht für „receive“. Zeit in Sekunden, für die Aufrechterhaltung des Empfang. „0“ bedeutet, es existiert kein Timeout.
- **Max. eingehende Verbindungen**: Anzahl der gleichzeitig eingehenden Verbindungen.
- **X.25 Facilities**

Spezielle Informationen oder Anlagen bei X.25-Übertragung

- **X.25 Benutzerdaten**
  Benutzerangaben für X.25-Übertragung
- **X.25 Fenstergröße**
  Die Fenstergröße bei X.25/ISDN-Kommunikation ist die Anzahl der Pakete, die ohne Bestätigung verschickt werden können. Die Fenstergröße kann während des Verbindungsaufbaus ausgehandelt werden. Wir empfehlen aber von vornherein die zum Partnernetzwerk passende Fenstergröße zu konfigurieren, z. B. den Wert „7“ für ISDN.
- **X.25 Geschlossene Benutzergruppe**
  ISDN und X.25 bieten die Möglichkeit, eine geschlossene Benutzergruppe zu bilden. Alle Teilnehmer, die zu einer solchen Gruppe gehören, können untereinander über das öffentliche Telekommunikationsnetz kommunizieren. Verbindungsanforderungen an Gruppenmitglieder von Teilnehmern, die nicht Mitglied der geschlossenen Benutzergruppe sind, werden von der Vermittlungsstelle abgewiesen. Standard: deaktiviert.

25.6.3.3 XOT Empfänger konfigurieren

XOT steht für X.25 over TCP. X.25 ist ein internationaler Telekommunikationsstandard, der von der Internationalen Fernmeldeunion verabschiedet wurde.

Weitere Informationen finden Sie unter www.itu.int

- **VectorPosition**: Nicht editierbar.
- **Empfänger-Nummer**: Nicht editierbar.
- **Lokale IP Adresse**: IP-Adresse oder Rechnername Ihres lokalen Systems.
- **Lokaler Port**: Port Ihres lokalen Systems, Standard: 1998.
- **Router IP Adresse**: IP-Adresse oder Rechnernamen Ihres ISDN-Routers.
- **Router Port**: Port Ihres ISDN-Routers, Standard: 1998.
- **X.25 Adresse**: Jeder Rechner, der in einem X.25 Netzwerk kommunizieren soll, muss eine X.25 konforme Adresse haben. Eine X.25 Adresse ist eine Folge von Ziffern. Die ersten Ziffern geben den Ländercode an, die darauffolgenden Ziffern bestimmen das Netzwerk innerhalb des Landes und die letzten Ziffern die Rechnernummer.
- **X.25 Facilities**: Meist nicht angegeben.
- **X.25 Benutzerdaten**: Meist nicht angegeben.

Wenn Sie „Facilities“ und „Benutzerdaten“ angeben möchten, dann als HEX-Code. Die Daten werden nur für X.25 über ISDN Übertragungen benötigt, wenn Ihr X.25 Provider dies fordert.

- **RvsTimeOut**: Zeit in Sekunden, nach welcher der Versuch, eine Verbindung aufzubauen, abgebrochen wird.
- **RvsRestartTimeOut**: Zeit in Sekunden, nach welcher der Versuch, eine Verbindung neu aufzubauen, abgebrochen wird.
- **X.25 Paketgröße**: Hier wird üblicherweise die Paketgröße 128 angegeben.
- **X.25 Fenstergröße**: Anzahl der Pakete, die ohne Bestätigung verschickt werden können, Standard: 7.
- **Max. eingehende Verbindungen**
  Maximale Anzahl gleichzeitig laufender Empfangsprozesse über diesen Kanal, Standard: 1, Maximum: 100.
- **X.25 DBit**: (Delivery Confirmation Bit) Wenn ausgewählt, dann fordert die sendende Station eine Empfangsbestätigung an.
- **X.25 Modulo**: Modulo-Variante, Standard: 8.
  → Siehe *X.25 Modulo (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 25, S. 263)*

25.6.3.4 TLS-Empfänger konfigurieren

Mit einem TLS-Empfänger (Transport Layer Security) kommunizieren Sie verschlüsselt.

- **IP-Adresse**
  IP-Adresse des TLS-Empfängers
- **Port**
  Port, über den der TLS-Empfänger erreichbar ist

26 OpenOffice Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Mit dem OpenOffice Connector können Sie Microsoft-Word-/OpenOffice-Writer-Dokumente automatisiert bearbeiten, konvertieren, auf entfernten Rechnern drucken (Remote Printing Service) und Dokumentvorlagen z. B. für Visitenkarten oder Abrechnungen mit Office pflegen.

Microsoft-Excel und OpenOffice-Calc-Dokumente können Sie automatisiert in andere Formate konvertieren.

→ Siehe *XML Schema als Vorlage für Eingangsnachrichten (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 26, S. 270)*

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

26.1 Voraussetzung: OpenOffice-Installation

- Eine OpenOffice.org-Installation mind. 3.3.0 ist von der inubit Suite 6 erreichbar
- OpenOffice ist im Servermodus gestartet:
  Dazu nutzen Sie aus dem OpenOffice-Programm-Verzeichnis `<OpenOfficeInstallDir>/App/openoffice/program` folgenden Befehl:

```
soffice[.exe] -headless -nofirststartwizard
-access
="socket,host=localhost,port=8100;urp;StarOffice
.Service"
```

26.2 Funktionsprinzip

Der OpenOffice Connector erwartet eine XML-Eingangsnachricht mit folgenden Elementen:

- Base64-kodiertes Eingangsdokument
- Ausgabefilter (optional)
- Operationen (Änderung, Konvertierung oder Drucken), die auf dem Dokument ausgeführt werden sollen.

Der OpenOffice Connector öffnet das angegebene Eingangsdokument und führt die Operationen aus. Abschließend erstellt er das Zieldokument in dem angegebenen Format oder sendet es an den konfigurierten Drucker.

XML Schema als Vorlage für Eingangsnachrichten

Die Eingangsnachrichten für den OpenOffice Connector erstellen Sie auf Basis des mitgelieferten XML Schemas `OOC.xsd` mit einem XSLT Converter.

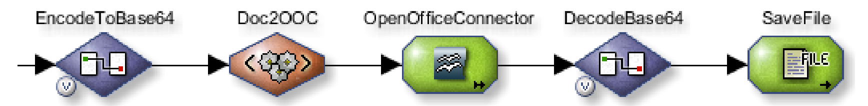
- Knoten OOWriter: Attribute für Word-/Writer-Dokumente
- Knoten OOCalc: Attribute für Excel-/Calc-Dokumente

Das XML Schema finden Sie im Repository in folgendem Verzeichnis:

```
/Global/System/Mapping Templates/OpenOffice
Connector
```

26.3 Beispiel-Workflow erstellen

Mit dem folgenden Beispiel-Workflow können Sie, je nach Konfiguration des XSLT Converters und des File Connectors, Word- und Writer-Dokumente bearbeiten, konvertieren oder drucken:



Assign-Modul zum Kodieren als Base64

So gehen Sie vor

1. Erstellen Sie ein Assign-Modul und benennen Sie es.
2. Erstellen Sie eine neue Abbildungsregel für das Variablen-Mapping.
   - Wählen Sie als Quelle „Ausgangsnachricht“ und als Datentyp „Binärdaten“.
   - Wählen Sie als Ziel „Variable“.

- Erstellen Sie die Variable „oo.document“ mit dem Typ „xs:base64Binary“.

3. Speichern Sie die Abbildungsregel mit „OK“.

XSLT Converter

So gehen Sie vor

1. Erstellen Sie einen XSLT Converter.
2. Aktivieren Sie auf der Seite „XSLT Converter Eigenschaften“ die Checkbox „Eingangsnachricht ignorieren“.
3. Öffnen Sie im Fenster „XML-Zieldatei“ die Schemadatei „OOC.xsd“

   → Siehe *XML Schema als Vorlage für Eingangsnachrichten (Workbench/Process Engine: Systemkonnektor-Guide, Kap. , S. 270).*.
4. Ziehen Sie das Root-Element „OpenOfficeConnector“ der Schemadatei aus dem Bereich „XML-Ziel“ auf das Element „xsl:template“ im Bereich XML-Ziel des XSLT Stylesheets.
5. Entfernen Sie alle nicht benötigten Elemente.
6. Fügen Sie mit dem XSLT-Kommandoassistenten das Element „param“ oberhalb des Elements „xls:output“ ein.
7. Weisen Sie dem Element „InputDocument“ das XSLT-Kommando „value-of“ und den Wert „$oo.document“ zu.

   Damit lesen Sie den Wert der im Assign-Modul definierten Variablen „oo.document“, die den Inhalt des zu bearbeitenden Dokuments enthält, in das Element „InputDocument“ ein.
8. Klicken Sie auf „Fertigstellen“.
9. Verbinden Sie das Assign-Modul mit dem XSLT Converter.

OpenOffice Connector einfügen

So gehen Sie vor

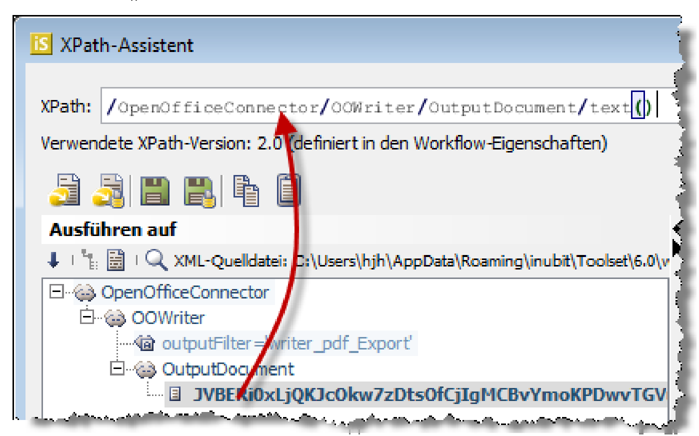
1. Erstellen Sie einen OpenOffice Connector.
2. Geben Sie an, auf welchem System der OpenOffice-Server installiert ist (Feld „Servernamen“) und über welchen Port er erreicht werden kann.
3. Verbinden Sie den XSLT Converter mit dem OpenOffice Connector.
4. Setzen Sie den Startpoint vor das Assign-Modul.
5. Starten Sie den Workflow mit einer Word-/Writer-Datei.
6. Öffnen Sie die Watchpoint-Datei nach dem OpenOffice Connector.
7. Speichern Sie die Watchpoint-Datei in die Zwischenablage.

Assign-Modul zum Dekodieren aus Base64 einfügen

So gehen Sie vor

1. Erstellen Sie ein Assign-Modul und benennen Sie es.

2. Erstellen Sie eine neue Abbildungsregel für das Variablen-Mapping.
   a. Wählen Sie als Quelle „Eingangsnachricht“ und als Datentyp „XML“.
   b. Klicken Sie auf den Button „XPath-Element auswählen“ am Ende der Zeile „XPath“.
   c. Öffnen Sie die Zwischenablage mit der beim Anlegen des OpenOffice Connectors im *Schritt 7* gespeicherten Watchpoint-Datei.
   d. Ziehen Sie den Textknoten des Elements „OutputDocument“ in die Zeile „XPath“.



   e. Speichern Sie die Änderungen mit „OK“.
   f. Wählen Sie als Ziel „Variable“.
   g. Erstellen Sie die Variable „oo.result“ mit dem Typ „xs:base64Binary“.
3. Erstellen Sie eine weitere Abbildungsregel für das Variablen-Mapping.
   a. Wählen Sie als Quelle „Variable“
   b. Wählen Sie als Variable die im *Schritt 2* erstellte Variable „oo.result“ aus.
   c. Wählen Sie als Ziel „Eingangsnachricht“ und als Kodierung „UTF-8“.
4. Speichern Sie die Abbildungsregel mit „OK“.
5. Verbinden Sie den OpenOffice Connector mit dem Assign Modul.

File Connector einfügen

So gehen Sie vor

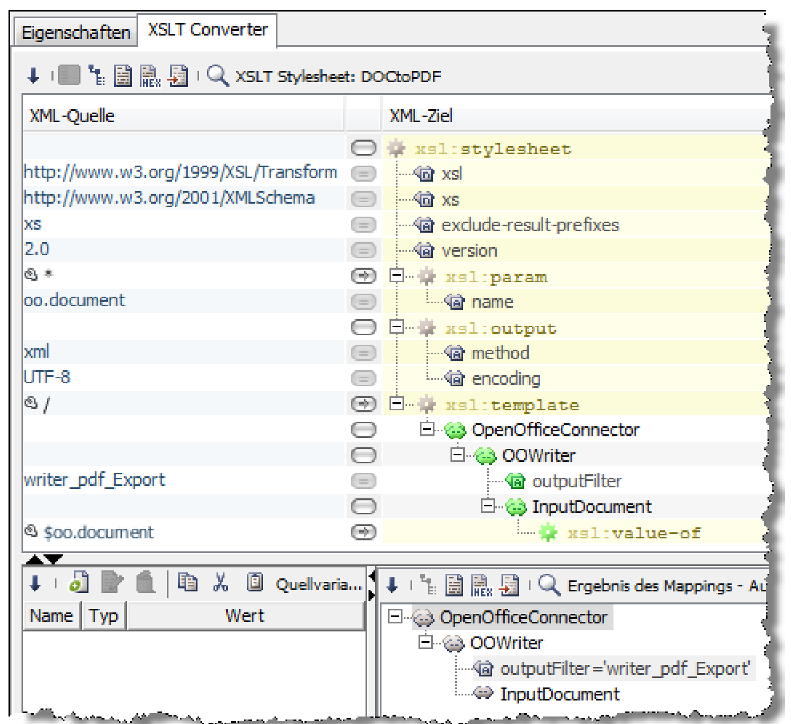
1. Erstellen Sie einen File Output Connector und benennen Sie ihn.
2. Wechseln Sie auf die Seite „Zu schreibende Datei“.
3. Wählen Sie als Eingangsformat „Daten“.
4. Geben Sie den Namen des Verzeichnisses ein, in dem die Zieldatei gespeichert werden soll.
5. Geben Sie den Namen der Zieldatei ein.

6. Verbinden Sie das Assign-Modul mit dem File Connector.

26.4 Dokument in das PDF-Format konvertieren

So gehen Sie vor

1. Erstellen Sie in einem XSLT Converter eine Eingangsnachricht.
   → Siehe *Beispiel-Workflow erstellen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 26.3, S. 270).*
2. Aktivieren Sie im Dialog „XSLT Converter Eigenschaften“ die Checkbox „Eingangsnachricht ignorieren“.
3. Passen Sie das XSLT Stylesheet so an, dass die Nachricht als Ergebnis des Mappings folgende Struktur hat:



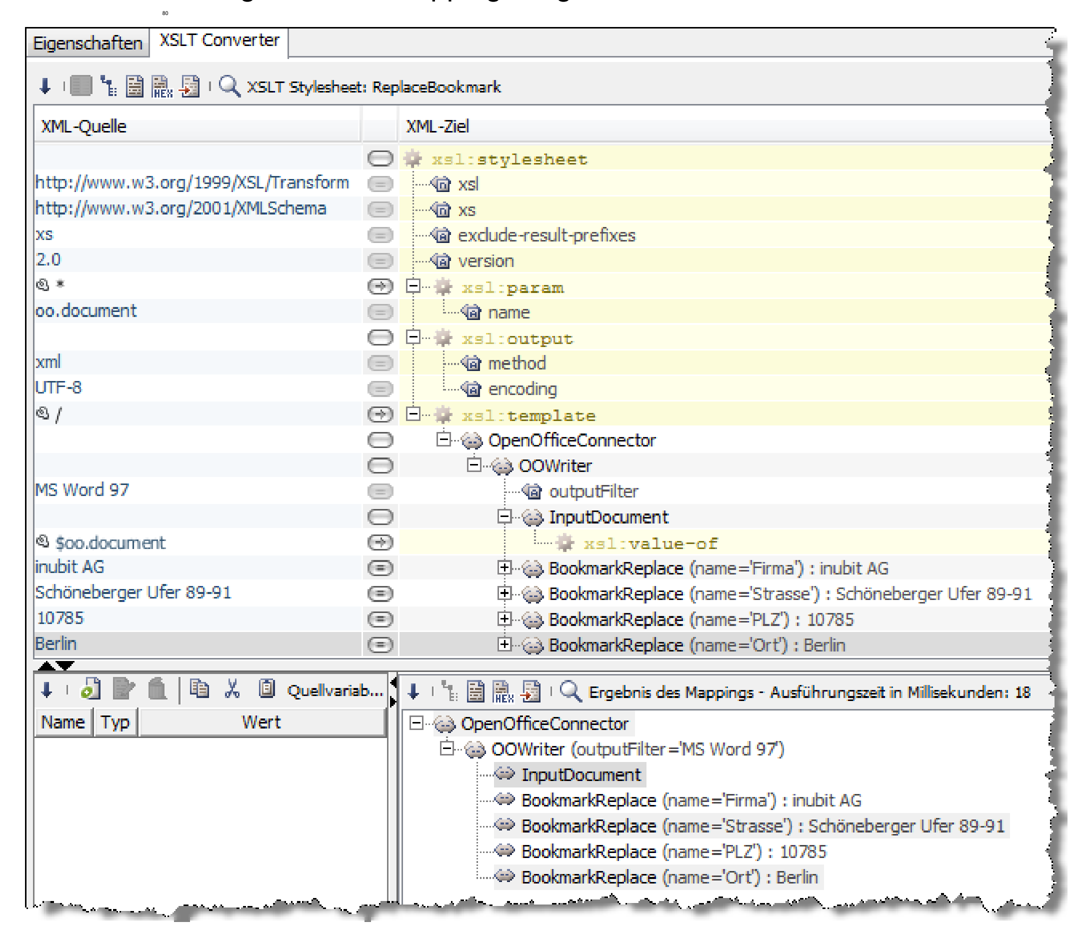
4. Weisen Sie dem Element „outputFilter“ im XSLT Stylesheet den Wert „writer_pdf_Export“ zu.
5. Ersetzen Sie den XSLT Converter im Beispiel-Workflow durch den im *Schritt 1* erstellten XSLT Converter.
6. Passen Sie im File Connector auf der Seite „Zu schreibende Datei“ ggf. den Dateinamen und das Verzeichnis an.
7. Setzen Sie den Startpoint vor das Assign-Modul am Anfang des Beispiel-Workflows.
8. Starten Sie den Workflow mit dem zu konvertierenden Dokument.

9. Prüfen Sie, ob das Dokument korrekt konvertiert wurde.

26.5 Dokument bearbeiten

So gehen Sie vor

1. Erstellen Sie ein Word- oder Writer-Dokument mit mehreren Textmarken, z. B. für eine Adresse.
2. Erstellen Sie mit einem XSLT Converter eine Eingangsnachricht.
   → Siehe *Beispiel-Workflow erstellen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 26.3, S. 270).*
3. Aktivieren Sie auf der Seite „XSLT Converter Eigenschaften“ die Checkbox „Eingangsnachricht ignorieren“.
4. Passen Sie das XSLT Stylesheet so an, dass die Nachricht als Ergebnis des Mappings folgende Struktur hat:



Je nach Art der gewünschten Änderungen können Sie einzelne BookmarkReplace-Elemente weglassen oder weitere Elemente gemäß der Schemadatei hinzufügen.

→ Refer to *XML Schema als Vorlage für Eingangsnachrichten (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 26, S. 270)*.

5. Ersetzen Sie den XSLT Converter im Beispiel-Workflow durch den im *Schritt 1* erstellten XSLT Converter.
6. Passen Sie im File Connector auf der Seite „Zu schreibende Datei“ ggf. den Dateinamen und das Verzeichnis an.
7. Setzen Sie den Startpoint vor das Assign-Modul am Anfang des Workflows.
8. Starten Sie den Workflow mit dem zu bearbeitenden Dokument.
9. Prüfen Sie, ob das Dokument korrekt bearbeitet wurde.

26.6 Dokument drucken

So gehen Sie vor

1. Erstellen Sie ein Assign-Modul, um das Eingangsdokument Base64 zu kodieren.

   → Siehe *Assign-Modul zum Kodieren als Base64 (Workbench/ Process Engine: Systemkonnektor-Guide, Kap. 26, S. 270)*.
2. Erstellen Sie mit einem XSLT Converter eine Eingangsnachricht.

   → Siehe *Beispiel-Workflow erstellen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 26.3, S. 270).*
3. Aktivieren Sie auf der Seite „XSLT Converter Eigenschaften“ die Checkbox „Eingangsnachricht ignorieren“.
4. Passen Sie das XSLT Stylesheet so an, dass die Nachricht als Ergebnis des Mappings folgende Struktur hat:

Als Attribute sind nur der Name des Druckers im Attribut „printerName“ und die zu druckende(n) Seite(n) im Attribut „pages“ zulässig.

Einzelne zu druckende Seiten trennen Sie durch ein Komma. Die erste und die letzte Seite eines Druckbereichs verbinden Sie durch einen Bindestrich.

5. Verbinden Sie das Assign-Modul mit dem XSLT Converter.
6. Erstellen Sie einen OpenOffice Connector.
7. Verbinden Sie den XSLT Converter mit dem OpenOffice Connector.
8. Setzen Sie den Startpoint vor das Assign-Modul.
9. Starten Sie den Workflow mit dem zu druckenden Dokument.
10. Prüfen Sie, ob das Dokument korrekt gedruckt wurde.

26.7 Dialog „Grundeinstellungen“

In diesem Dialog haben Sie folgende Optionen:

OpenOffice Server

- **Servername**

  IP-Adresse oder Host-Name der Systems, auf dem OpenOffice installiert ist.

- **Port**
  Port, über den OpenOffice Connector erreicht wird.
- **Standard**
  Setzt den Port auf den Standardwert 8100 zurück.

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

27 OSCI Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Online Service Computer Interface (OSCI) ist ein Nachrichten-Standard und dient als Grundlage für die rechtssichere Datenübertragung im eGovernment-Bereich auf Basis von digitaler Signatur und Verschlüsselung zwischen zwei Kommunikationspartnern über eine virtuelle Poststelle (Intermediär). Die Daten werden in Form von OSCI-Nachrichten übertragen.

Der OSCI Connector verbindet sich mit einer virtuellen Poststelle und ermöglicht es, innerhalb eines Workflows

- OSCI-Nachrichten von einer virtuellen Poststelle entgegenzunehmen, zu entpacken, zu entschlüsseln und ihre Signatur zu prüfen.
- OSCI-Nachrichten für eine virtuelle Poststelle zu erstellen, zu signieren und zu verschlüsseln.

Konnektortypen

Die Funktionen des OSCI Connectors hängen von der konkreten Konfiguration ab:

- **Input Connector**

  Ruft OSCI-Nachrichten im eingestellten Zeitintervall von einer virtuellen Poststelle ab und entpackt diese. Der Input Connector wird nur aktiv, wenn der Scheduler aktiviert ist.

  → Siehe *Dialog „Zeitgesteuerte Verarbeitung“ (Workbench/ Process Engine: Systemkonnektor-Guide, Kap. 1.1.3, S. 22).*

- **Medium Connector**

  Versendet verschlüsselte und signierte OSCI-Nachrichten an eine Virtuelle Poststelle und leitet die Response an das nachfolgende Modul im Workflow weiter.

Die aktuelle OSCI-Spezifikation finden Sie unter http://www1.osci.de/sixcms/media.php/13/osci_spezifikation_1_2_deutsch.pdf.

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

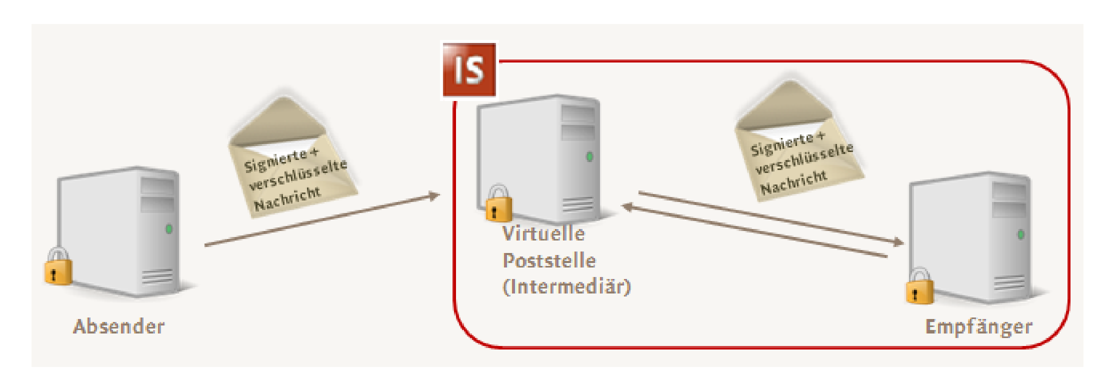
27.1 Voraussetzungen

- Adresse (URL) des Intermediärs ist vorhanden
- Öffentliches Schlüssel-Zertifikat *cry.cer des Intermediärs ist vorhanden
- Privates Signatur-Zertifikat ist vorhanden
- Privates Schlüssel-Zertifikat ist vorhanden
- Passwörter für die privaten Zertifikate sind vorhanden

Speichern Sie die Zertifikatsdateien in einem entsprechenden Verzeichnis. Die Zertifikate für die privaten Schlüssel sind `*.pfx`-Dateien, die für die öffentlichen Schlüssel `*.cer`-Dateien.

27.2 Funktionsprinzip

Sie können einen OSCI Connector verwenden, um OSCI-Nachrichten, die zwischen einem OSCI-Client als Absender, einer virtuellen Poststelle (VPS) als OSCI-Intermediär und einem Empfänger transportiert werden, mit der inubit Suite 6 über den Intermediär zu empfangen oder über ihn zu versenden.



Datenübermittlung

OSCI ist der eGovernment-Standard für die sichere Datenübermittlung. Dabei werden die Daten in Form von OSCI-Nachrichten versendet und empfangen.

Zertifikatsverwaltung

Zum Verschlüsseln und Signieren von OSCI-Nachrichten werden Zertifikate benutzt, die gleichzeitig zur Adressierung der Nachrichten verwendet werden.

Kommunikationsablauf

Die zu übermittelnden Daten werden in Form von Nachrichten zwischen den Kommunikationspartnern ausgetauscht. Dabei können die Kommunikationspartner sowohl Absender als auch Empfänger von Nachrichten sein. Die zu einer Nachricht gehörenden Daten werden nach ihrer Erfassung vom Absender elektronisch signiert und dann im OSCI-Format an einen so genannten Intermediär gesendet.

Der Intermediär übernimmt die Funktionalitäten einer virtuellen Poststelle (VPS)

- prüft die Signatur und die Zertifikate,
- erstellt darüber ein Prüfprotokoll und
- hält die Nachricht im Server-Postfach des Empfängers zum Abruf bereit.

Der OSCI-Intermediär wird von der virtuellen Poststelle für die Verarbeitung von OSCI-Nachrichten genutzt. Dabei ist die virtuelle Poststelle eine Lösung, um verschlüsselte und signierte Nachrichten, die eine Behörde in den verschiedensten Formaten und Protokollen erhält, entgegenzunehmen, zu konvertieren und zu verarbeiten.

Der OSCI Connector verbindet sich mit der virtuellen Poststelle und kann mit den entsprechenden Zugangsdaten OSCI-Nachrichten empfangen sowie an diese senden.

Das entsprechende Schema-Template finden Sie im iS-Repository unter „Global > System > Mapping Templates > OSCI Connector“.

27.3 Dialogbeschreibungen

Dieser Abschnitt erläutert die folgenden Themen:

- *Dialog „Abrufen von OSCI-Nachrichten am Intermediär“, S. 281*
- *Dialog „Erstellen von OSCI-Nachrichten am Intermediär“, S. 282*

27.3.1 Dialog „Abrufen von OSCI-Nachrichten am Intermediär“

(Input Connector)

In diesem Dialog legen Sie die Konfigurationseinstellungen des Empfängers fest, um OSCI-Nachrichten am Intermediär abzurufen.

Da der Input Connector nur Nachrichten abruft, wenn der Scheduler aktiviert ist, müssen Sie diesen im *Dialog „Zeitgesteuerte Verarbeitung“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1.1.3, S. 22)* aktivieren, um das Polling-Intervall festzulegen.

Virtuelle Poststelle

- **Intermediär**
  Geben Sie die URL des Intermediärs, unter der dieser online erreichbar ist.
- **Zertifikat des Intermediärs**
  Laden Sie hier die Datei mit dem öffentlichen Zertifikat des Intermediärs; entspricht einer `*.cer`-Datei.

Empfänger

- **Signatur-Keystore**
  Laden Sie hier die `*.pfx`-Datei mit dem privaten Schlüssel des Empfängers.
- **Passwort des Signatur-Schlüssels**
  Geben Sie das Keystore-Passwort des privaten Schlüssels für die Signaturprüfung ein.
- **Chiffrierungs-Keystore**
  Laden Sie hier die `*.pfx`-Datei mit dem privaten Schlüssel für die Entschlüsselung ein.
- **Passwort des Chiffrierungs-Schlüssel**
  Geben Sie das Keystore-Passwort des privaten Schlüssels für die Entschlüsselung ein.

Verbindungstest

Verbindung testen

Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

27.3.2 Dialog „Erstellen von OSCI-Nachrichten am Intermediär“

(Medium Connector)

In diesem Dialog legen Sie die Konfigurationseinstellungen des Autors und Empfängers fest, um OSCI-Nachrichten am Intermediär zu erstellen.

Virtuelle Poststelle

- **Intermediär**

Geben Sie die URL des Intermediärs, unter der dieser online erreichbar ist.

- **Zertifikat des Intermediärs**
  Laden Sie hier die Datei mit dem öffentlichen Zertifikat des Intermediärs; entspricht einer `*.cer`-Datei.

Autor

- **Signatur-Keystore**
  Laden Sie die `*.pfx`-Datei mit dem Zertifikat des Absenders, das zum Signieren dient.
- **Passwort des Signatur-Schlüssels**
  Geben Sie das Keystore-Passwort des privaten Schlüssels für die Signierung der erstellten Nachricht ein.
- **Chiffrierungs-Keystore**
  Laden Sie hier die `*.pfx`-Datei mit dem Zertifikat des privaten Schlüssels des Absenders zur Verschlüsselung der erstellten Nachricht ein.
- **Passwort des Chiffrierungs-Schlüssels**
  Geben Sie das Keystore-Passwort des privaten Schlüssels für die Verschlüsselung der erstellten Nachricht ein.

Empfänger

- **Chiffrierungs-Zertifikat**
  Laden Sie die Datei mit dem öffentlichen Zertifikat des Empfängers.

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

28 REST Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Der REST Connector bietet eine Schnittstelle, um Daten via HTTP zu übertragen, ohne dass es dabei eine zusätzliche Transportschicht wie z.B. SOAP gibt. Der REST Connector wird genutzt, um REST-konforme Web Services über eine URL aufzurufen oder anzubieten.

Konnektortypen

Die Funktionen des REST Connectors hängen von der konkreten Konfiguration ab:

- **Input Listener Connector**

  Stellt eine Ressource unter einer angegebenen URL zur Verfügung, die von einem Client per Request angefragt werden kann. Die angebotene Ressource stellt dabei das Ergebnis einer Workflow-Ausführung dar.

  Alle Parameter und Header, die mit dem Request übergeben wurden, stehen während der Workflow-Ausführung als Modulvariablen zur Verfügung.
- **Medium/Output Connector**

  Sendet als Client HTTP-Requests an einen HTTP-Server und gibt - im Fall des Medium Connectors - die zurückgelieferte Antwort (falls vorhanden) an den Workflow weiter.

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

28.1 Modulvariablen des REST Connectors

Bei der Ausführung eines REST Connectors werden abhängig vom Konnektor-Typ folgende Variablen gesetzt und stehen im weiteren Verlauf der Workflow-Ausführung zur Verfügung:

Input Listener Connector

Für die Daten der empfangenen Anfrage werden folgende Variablen gesetzt:

- `restConnector.requestMethod`
  Enthält den Namen der HTTP-Methode der Anfrage.
- `restConnector.requestHeaders`
  Enthält die Header-Daten der Anfrage-Nachricht.
- `restConnector.requestQueryParams`
  Enthält die Werte der Query-Parameter aus der Anfrage-URL. Diese werden in eine XML-Struktur überführt und jeder Parameter wird in ein XML-Element umgesetzt.
- `restConnector.requestResourcePath`
  Enthält den Pfad der angeforderten Ressource. Die Variable enthält z. B. bei der Anfrage-URL `http://localhost:8000/ibis/rest/rc/orders/123` den Wert `/orders/123`.
- `restConnector.requestURITemplateParam.x`
  Enthält die Inhalte von dynamischen Pfadbestandteilen der Anfrage-URL. „x“ steht dabei jeweils für den in geschweiften Klammern stehenden Namen des Pfadbestandteils innerhalb der am Connector konfigurierten Ressourcen-URL. So werden z. B. bei der Ressourcen-URL `users/{userName}/auftraege/{auftragsNr}` die Variablen `restConnector.requestURITemplateParam.userName` und `restConnector.requestURITemplateParam.auftragsNr` mit den konkreten Werten aus der Anfrage-URL gesetzt.
- `restConnector.clientAddress`
  Enthält die IP Adresse des aufrufenden Client.
- `restConnector.requestAuthUser`
  Enthält den Namen des authentifizierten Benutzers, falls bei den Authentifizierungs-Optionen „Portal-Benutzerdaten verwenden“ ausgewählt wurde.

Die Status- und Header-Daten der zu versendenden Antwort werden in folgende Variablen geschrieben:

- `restConnector.responseStatusCode`
  Enthält Antwortcode für das Ergebnis der gesendeten Antwort.
- `restConnector.responseStatusDescription`
  Enthält die Beschreibung des Antwortcodes.
- `restConnector.responseHeaders`

Enthält die Header-Daten der gesendeten Antwort-Nachricht.

Der Inhalt der Variablen entspricht den im Konnektor konfigurierten Standardwerten. Die Variablen dienen dazu, im Workflow dynamisch überschrieben zu werden. Der Wert der Variable, der am Ende des Workflows zur Verfügung steht, wird für den Versand der Antwort verwendet.

Medium/Output Connector

Die Status- und Header-Daten der empfangenen Antwort werden in folgende Variablen geschrieben:

- `restConnector.responseStatusCode`
  Liefert den Antwortcode, der das Ergebnis der Abfrage angibt.
- `restConnector.responseStatusDescription`
  Liefert die Beschreibung des Antwortcodes, z. B. „Internal Server Error“.
- `restConnector.responseHeaders`
  Liefert die Header-Daten der Antwort-Nachricht.

→ Siehe *Workflow-Variablen und Mappings (Workbench: Benutzer-Guide, Kap. 14, S. 365).*

28.2 REST-Funktionsprinzip

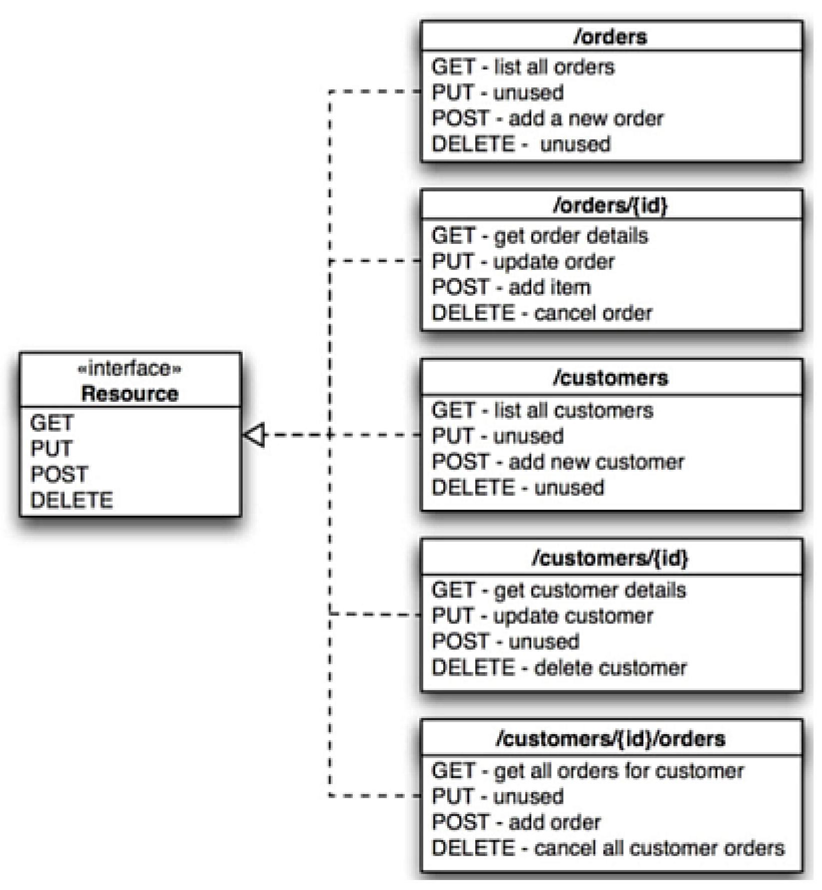
HTTP - Ressourcen - URLs

REST (Representational State Transfer) ist ein Architektur-Konzept für Web Services, das beschreibt, wie Web-Standards in einer Web-gerechten Weise einsetzt werden können. REST-basierte Web Services nutzen das HTTP-Protokoll und die HTTP-Methoden wie GET oder POST.

Ein Grundprinzip dieses Modells ist, dass die Funktionalität eines Web Service nicht in dessen Eigenschaften abgebildet wird, sondern in Ressourcen des Service. Die einzelnen Funktionen eines Dienstes werden dabei in einzeln über URLs adressierbare Ressourcen zerlegt.

REST bietet damit die Möglichkeit, Applikationen wie z. B. einen Online-Shop über eine URL zu adressieren. Dabei stellt jedes einzelne Objekt der Anwendung wie Artikel oder Kunde eine Ressource dar, die über eine eigene URL angesprochen wird.

Jede REST-Ressource besitzt über die verschiedenen HTTP-Methoden eine generische Schnittstelle.



Für das Beispiel eines Online-Shops als REST-Service gibt es bei der Auftragsverwaltung z. B. eine URL für die Menge der Bestellungen jedes Kunden und eine URL für jede Bestellung.

Die Kommunikation erfolgt nur bei Bedarf auf Abruf. Der Client bzw. Konnektor wird aktiv und fordert vom Server eine Repräsentation einer Ressource an bzw. modifiziert eine Ressource.

Ressourcen

Im Beispiel des Online-Shops stellt dieser eine Ansammlung von Ressourcen dar, an die per HTTP Nachrichten gesendet werden können, z. B. zum Aufruf eines Kunden zu einer Bestellung in einem Warenkorb. Die Ressource wird dabei nicht direkt manipuliert, denn der Zugriff erfolgt indirekt über die der Ressource zugeordnete URL.

Beispiele für Ressourcen

Online-Shop als Web Service:

`http://localhost:8000/ibis/rest/sqlshop`

Auflistung aller Aufträge:

`http://localhost:8000/ibis/rest/sqlshop/orders`

Auftrag mit der Nr. 1234:

`http://localhost:8000/ibis/rest/sqlshop/orders/1234`

Repräsentationen

Jede Ressource kann verschiedene Repräsentationen haben, d.h. der Kunde kann z. B. durch eine XML-Struktur mit den Kundenstammdaten oder durch eine Visitenkarte im vcard-Format repräsentiert sein. Zwischen Client und Server muss ein gemeinsames Verständnis über die Bedeutung der Repräsentation durch die Verwendung von XML vorhanden sein.

Repräsentationen können auf weitere Ressourcen verweisen, die ihrerseits wieder Repräsentationen liefern, die wiederum auf Ressourcen verweisen können. Wenn ein Client einem Link in einer Repräsentation folgt, gelangt er zu einer anderen Ressource.

Beispiele für Repräsentationen

Repräsentation eines Kunden durch seine Stammdaten:

```
<CUSTOMER>
  <ID>4</ID>
  <FIRSTNAME>Heinz</FIRSTNAME>
  <LASTNAME>Mustermann</LASTNAME>
  <STREET>Musterstraße. 13</STREET>
  <CITY>Musterstadt</CITY>
</CUSTOMER>
```

Repräsentation einer Kundenliste mit Verlinkungen auf die einzelnen Kunden als weitere Ressourcen:

```
<CUSTOMERList>
  <CUSTOMER xlink:href="http://www.inubit.com/
sqlrest/CUSTOMER/1/">1</CUSTOMER>
  <CUSTOMER xlink:href="http://www.inubit.com/
sqlrest/CUSTOMER/2/">2</CUSTOMER>
  <CUSTOMER xlink:href="http://www.inubit.com/
sqlrest/CUSTOMER/3/">3</CUSTOMER>
  <CUSTOMER xlink:href="http://www.inubit.com/
sqlrest/CUSTOMER/4/">4</CUSTOMER>
  <CUSTOMER xlink:href="http://www.inubit.com/
sqlrest/CUSTOMER/29/">29</CUSTOMER>
</CUSTOMERList>
```

Methoden

Das REST-Prinzip sieht vor, dass auf Ressourcen mit den Standard-HTTP-Methoden GET, POST, PUT und DELETE zugegriffen wird. Dadurch müssen keine Protokoll-Konventionen bekannt sein, damit Client und Server kommunizieren können. Eine Anfrage (Request), z. B. nach einer bestimmten Bestellung, besteht immer aus einer URL und einer HTTP-Methode. Die HTTP-Methoden bilden dabei die grundlegenden Aktionen auf den Ressourcen ab.

→ Siehe *HTTP-Methoden verwenden (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 28.3, S. 290).*

28.3 HTTP-Methoden verwenden

Die in den REST Connectoren konfigurierbaren HTTP-Methoden decken die Anwendungsfälle von REST-Anwendungen ab.

HTTP-Methode	Bedeutung
GET	GET fragt die Repräsentation einer Ressource ab. GET-Requests können beliebig oft abgeschickt werden.
POST	Mit POST kann einer Ressource etwas hinzugefügt werden. Beispielsweise könnte eine Ware zu einem Warenkorb hinzugefügt werden.
PUT	Neue Ressourcen können mit PUT erzeugt oder der Inhalt bestehender Ressourcen kann mit PUT ersetzt werden.
DELETE	Ressourcen können mit DELETE gelöscht werden.
OPTIONS	Liefert eine Liste mit den vom Server unterstützten Methoden.
HEAD	Genau wie bei GET wird eine Ressource angefragt, es wird jedoch in der Antwort kein Inhalt (Body) übertragen, sondern nur der Response Header. So kann z. B. vor einer GET-Anfrage die Dateigröße abgefragt werden.

Um verschiedene Methoden auf eine Ressource anzuwenden, wird immer die gleiche URL verwendet, denn die URL ist unabhängig von der durchgeführten Aktion.

28.4 Ressource anbieten

Prinzip

Einen Web Service in Form einer REST-Ressource können Sie durch einen Technical Workflow mit einem REST Input Listener Connector realisieren. Der REST Connector bietet dann nach außen eine Ressource (z. B. eine Auftrags- oder Artikelliste) in Form einer URL an. Einzelne Ressourcen werden jeweils über einen einzelnen REST Input Connector angeboten.

Die Ressourcen sind einzelne Objekte einer Anwendung, die über eine URL erreichbar sein müssen und mit einem Dokument, vorzugsweise in XML, repräsentiert werden können.

So gehen Sie vor

1. **Ressource konfigurieren**
   a. Erstellen Sie einen REST Input Listener Connector.
   b. Wählen Sie aus der Liste unter „Vorgaben verwenden für“, welche Art von Ressource dieser Konnektor anbieten soll.
      → Siehe *Bereitgestellte Ressource (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 28, S. 296)*
   c. Definieren Sie eine URL, unter der auf die Ressource zugegriffen werden kann.

Für dynamische Pfadbestandteile können geschweifte Klammern verwendet werden, z. B. `http://localhost:8000/ibis/rest/orders/{orderID}`. Hier wird der Wert von „orderID“ als Workflow-Variable im Workflow zur Verfügung gestellt und kann benutzt werden, um die Daten des gewünschten Auftrags zu ermitteln. Geben Sie „*“ am Ende des Pfades ein, um Pfade zu unterstützen, die mit beliebigen Zeichen enden.

   d. Sichern Sie Ihren Rest-Service vor unzulässigen Zugriffen durch Aktivieren der Option „Authentifizierung erforderlich“.
   e. Legen Sie die gewünschte Authentifizierungsmethode fest und geben Sie die für einen Request erforderlichen Zugangsdaten an.
   f. Legen Sie fest, welche Operationen auf Ihrer Ressource ausgeführt werden können.

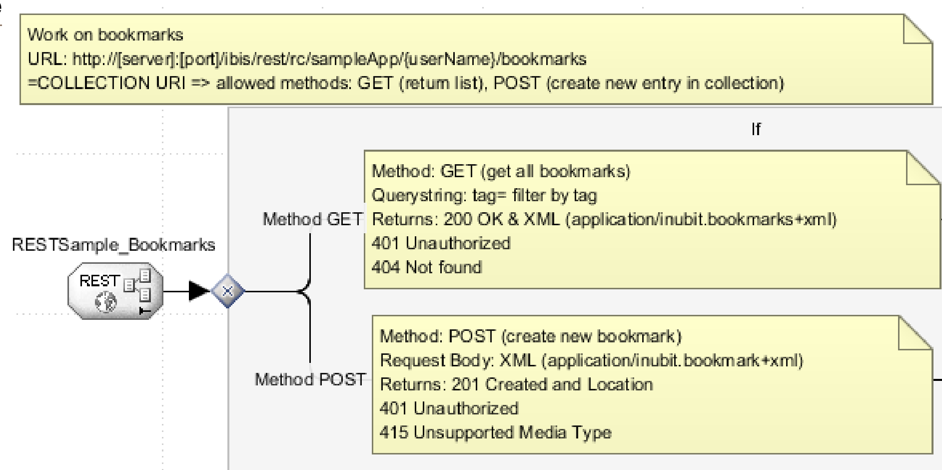
Es sind nicht für jeden Ressourcentyp alle HTTP-Methoden zugelassen. Standardmäßig sind die für eine gewählte Ressource sinnvollen HTTP-Methoden vorausgewählt.

   g. Legen Sie die Kodierung für Daten fest, die als Request im Format `multipart/form-data` eingehen.

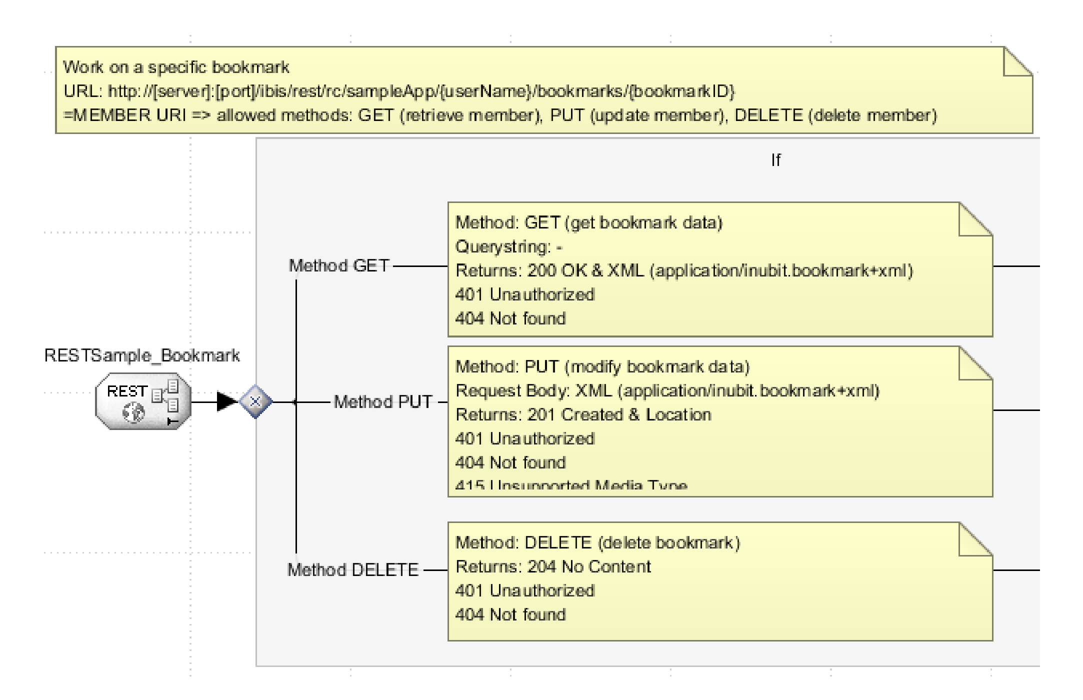
2. **Antwort konfigurieren**
   Sie definieren für jede zulässige HTTP-Methode, mit der die Ressource angefragt werden kann, die von der Ressource zurückgelieferte Antwort.
   a. Konfigurieren Sie für die verfügbaren Anfrage-Typen die Antwortdaten.
      → Siehe *Dialog „Konfiguration der Antwort“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 28.8.2, S. 298).*
      Die hier konfigurierten Werte können im Workflow dynamisch über das Ändern entsprechender Variablen geändert werden.
      → Siehe *Modulvariablen des REST Connectors (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 28.1, S. 286).*

Bei Ressourcen mit mehreren unterstützten Methoden ist die Variable `restConnector.requestMethod` besonders wichtig, da anhand des Wertes dieser Variable im Workflow verzweigt werden kann, um die jeweilige Methode zu implementieren.

Beispiel für Sammel-Ressource



Beispiel für Einzel-Ressource aus Sammlung



Bei eingehenden Requests mit HTML-Form-Daten und dem Content-Type `application/x-www-form-urlencoded` werden die Form-

Daten automatisch in ein XML-Dokument gesetzt. Dieses XML-Dokument wird zur Eingangsnachricht des Workflows.

28.5 Verfügbare Input Connectoren anzeigen

Sie können sich eine Liste aller in inubit Workbench erstellten, verfügbaren, aktiven und inaktiven REST Input-Konnektoren anzeigen lassen.

So gehen Sie vor

1. Wählen Sie einen REST Input Listener Connector aus und öffnen Sie diesen zum Bearbeiten.
2. Klicken Sie im „*Dialog „Konfiguration der Ressource“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 28.8.1, S. 296)*“ im Bereich „Bereitgestellte Ressourcen“ neben der URL auf .
3. Wählen Sie die Option „Alle REST Input-Konnektoren auflisten“.
   Ein Dialog öffnet sich, das eine Liste aller angelegten Input Connectoren anzeigt. In der Liste wird der Name des Konnektors, der entsprechende Workflow, der Pfad für die Ressource und die für den Konnektor verfügbaren HTTP-Methoden aufgeführt.

28.6 Publizierte Ressourcen anzeigen

Sie können sich eine Liste mit allen publizierten und aktivierten Ressourcen anzeigen lassen.

So gehen Sie vor

1. Wählen Sie einen REST Input Listener Connector aus und öffnen Sie diesen zum Bearbeiten.
2. Klicken Sie im *Dialog „Konfiguration der Ressource“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 28.8.1, S. 296)* im Bereich „Bereitgestellte Ressourcen“ neben der URL auf .
3. Wählen Sie die Option „Publizierte Ressourcen anzeigen“. Im Browser öffnet sich eine Seite, die aufbereitete Liste mit allen aktivierten Ressourcen anzeigt.

Ressourcen, die mit einem REST Input Listener Connector realisiert sind, werden erst als publiziert und aktiviert angezeigt, wenn die entsprechenden Technical Workflows aktiviert sind.

28.7 Ressource aufrufen

Prinzip

Sie können mit dem REST Connector eine Ressource, die über eine bestimmte URL zur Verfügung steht, aufrufen. So können Sie z. B. eine Artikelliste anfordern. Sie müssen dazu einen REST Output oder Medium Connector zum Versenden der Anfrage und zum Empfangen der Antwort anlegen und konfigurieren.

- **Medium Connector**
  Verwenden Sie den Medium Connector, wenn die aufgerufene Ressource eine Antwort zurücksendet, die im Workflow weiterverarbeitet werden soll.
- **Output Connector**
  Verwenden Sie den Output Connector, wenn die Anfrage an eine Ressource ein Aufruf ohne Rückgabe ist.

Voraussetzungen

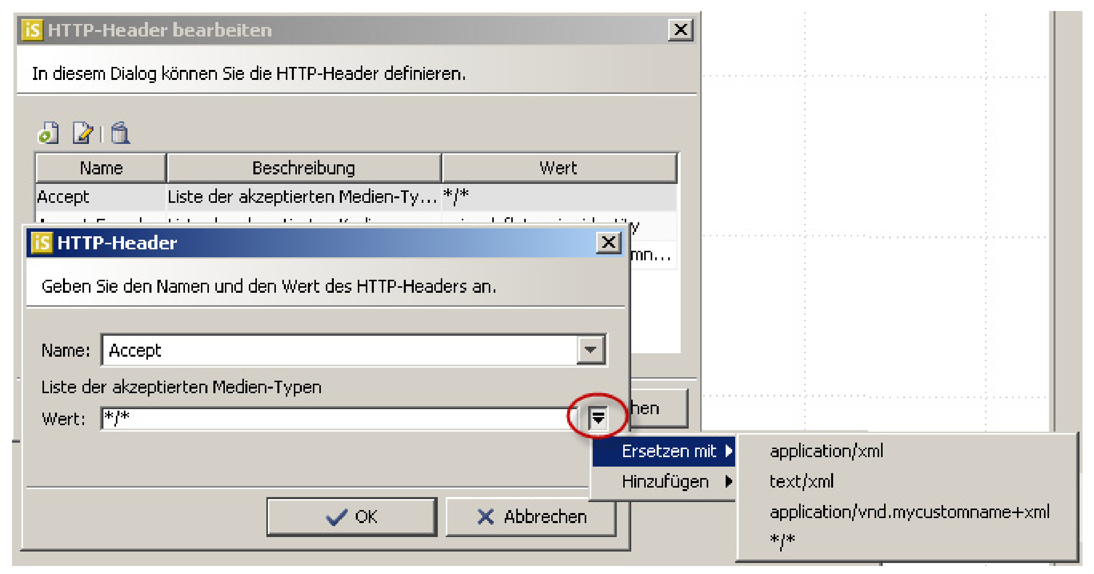
- URL der aufzurufenden Ressource ist verfügbar.
- Authentifizierungsdaten sind verfügbar, falls der Server für die Ressource eine Authentifizierung fordert.

So gehen Sie vor

1. Legen Sie einen neuen REST Medium oder Output Connector.
2. Geben Sie die URL der aufzurufenden Ressource ein.
3. Geben Sie ggf. die erforderlichen Authentifizierungsdaten ein.
   → Siehe *Authentifizierung verwenden (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 28, S. 300)*
4. Wählen Sie je nach Art der Anfrage die gewünschte HTTP-Methode.
   → Siehe *HTTP-Methoden verwenden (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 28.3, S. 290).*
5. Falls Sie als Methode PUT oder POST gesetzt haben, wählen Sie jeweils eine Option aus der Liste der Medientypen und des Zeichensatzes aus.
   → Siehe *Anfragedaten (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 28, S. 301)*

Damit legen Sie die Definition der Daten fest, die verschickt werden. Die Werte der Anfragedaten werden in den Content-Type-Header übernommen.

6. Optional können Sie Anpassungen im Request-Header vornehmen, z. B. wenn Sie für Ressourcen, die mehrere Repräsentationen unterstützen, die gewünschte Repräsentation der angefragten Ressource festlegen wollen. Dazu nutzen Sie den `Accept`-Header:
   a. Klicken Sie auf den Button „Header bearbeiten“. Ein Dialog öffnet sich. Standardmäßig wird mit */* eine beliebige Repräsentation einer Ressource abgefragt.
   b. Doppelklicken Sie den `Accept`-Header. Ein weiterer Dialog öffnet sich:



   c. Wählen Sie einen Medien-Typen aus. Sie können auch mehrere Typen auswählen und es wird dann eine geordnete Liste der gewünschten Repräsentationen zurückgeliefert.
   d. Schließen Sie die Dialoge zum Bearbeiten des HTTP-Headers mit „OK“.
7. Klicken Sie „Fertig stellen“.
8. Publizieren Sie den REST Connector.

Für das Abfragen einer Ressource mit GET benötigen Sie keine Ausgangsnachricht, da die Ressource nur angefragt, jedoch keine Änderung an den Daten vorgenommen werden. Für das Zugreifen auf eine Ressource mit POST oder PUT benötigen Sie eine Ausgangsnachricht. Falls Sie z. B. Daten in eine Artikelliste

hinzufügen möchten, können Sie mit einem XSLT Converter im Workflow vor dem REST Connector eine XML-Nachricht erstellen und diese mit versenden, z. B.:

```
<artikel>
  <beschreibung>Rooibusch-Tee</beschreibung>
  <preis>2,80</preis>
  <einheit>100g</einheit>
</artikel>
```

Alle Eingangsnachrichten in den REST Output/Medium Connector werden mit der Anfrage versendet.

Sie können die aufzurufende URL der Ressource dynamisch anpassen, indem Sie die Moduleigenschaft `Request.URI` mit Hilfe des Variablen-Mappings überschreiben siehe *Workflow-Variablen und Mappings (Workbench: Benutzer-Guide, Kap. 14, S. 365)*.

28.8 Dialogbeschreibungen

Dieser Abschnitt erläutert die folgenden Themen:

28.8.1 Dialog „Konfiguration der Ressource“

(Input Listener Connector)

In diesem Dialog legen Sie den Pfad für die vom Konnektor angebotene Ressource, die Authentifizierung und die HTTP-Methoden fest.

Bereitgestellte Ressource

- **Vorgaben verwenden für**

Wählen Sie aus, welchen Typ von Ressource der Konnektor anbieten soll. Die Wahl des Ressourcen-Typs legt gleichzeitig die Vorauswahl der standardmäßig unterstützten HTTP-Methode(n) fest.

→ Siehe *Unterstützte HTTP-Methoden (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 28, S. 298)*

- **Einzelne Ressource**

  Wählen Sie die Option, wenn nur lesend auf eine einzelne Ressource zugegriffen werden soll, z. B. „Bestellung anzeigen". Es wird automatisch die HTTP-Methode `GET` vorausgewählt.
- **Sammel-Ressource**

  Wählen Sie die Option, wenn die Ressource z. B. eine Liste von Aufträgen („/auftraege") umfassen soll und diese Liste abgerufen oder der Liste neue Einträge hinzugefügt werden sollen. Es sind standardmäßig nur die Methoden GET und POST erlaubt.
- **Einzel-Ressource aus Sammlung**

  Wählen Sie diese Option, wenn Sie z. B. einen einzelnen Auftrag aus einer Auftragsliste oder eine Artikelnummer als Ressource anbieten wollen. Unterstützte Methoden sind hier GET (Daten abrufen), PUT (Daten ändern) und DELETE (Daten löschen).

■ **URL**

URL des Input Listeners, unter welcher dieser auf Requests wartet und seine Ressource zur Verfügung stellt. Die URL ist standardmäßig nach folgendem Muster vorbelegt:

`http(s)://[ServerName]:[Port]/ibis/rest/rc/[NamedesInputListeners]`

Passen Sie ggf. den Teil der URL hinter `rc` beliebig an. Für dynamische Pfad-Bestandteile können geschweifte Klammern verwendet werden, z. B. /users/{userName}/auftraege/{auftragsNr}.

Geben Sie „*" am Ende des Pfades ein, um Pfade zu unterstützen, die mit beliebigen Zeichen enden.

Authentifizierung erforderlich

Markieren Sie diese Option, wenn für den Zugriff auf die Ressource eine Authentifizierung erforderlich sein soll.

■ **Statische Zugangsdaten verwenden**

Definieren Sie hier ein zu verwendendes Verfahren und die für den Zugriff erforderlichen Authentifizierungsdaten.

- **Verfahren**
  - **Basic:** Es werden Benutzername und Passwort verlangt und unverschlüsselt übermittelt.

Nur zu verwenden, wenn die Verbindung zwischen Client und Server als sicher betrachtet werden kann!

  - **Digest**: Verlangt ebenfalls Benutzername und Passwort. Das Passwort wird verschlüsselt übertragen. Für die Verschlüsselung werden mehrere Parameter verwendet, u. a. zufällig erzeugte Werte.
- **Benutzername**: Benutzername für die Authentifizierung.
- **Passwort**: Passwort für die Authentifizierung

- **Portal-Benutzerdaten verwenden**

  Diese Option ist nur bei einem installierten Portal verfügbar.

  Wenn die Option aktiviert ist, ist der Zugriff auf die Ressource nur möglich, wenn gültige Portal-Benutzerdaten übergeben werden.

  Als Authentifizierungs-Verfahren wird „Basic“ verwendet.

  Der in einem Aufruf verwendete Portal-Benutzername steht bei Workflow-Ausführung in der Variable `restConnector.requestAuthUser` zur Verfügung.

  → Siehe *Modulvariablen des REST Connectors (Workbench/ Process Engine: Systemkonnektor-Guide, Kap. 28.1, S. 286).*

- **Authentifizierung für Zugriff auf Process Engine verwenden (Passwort aus Portalkonfiguration)**: Verwendung des internen Benutzers (inubitISPortalUser@@@intern)

Unterstützte HTTP-Methoden

Hier legen Sie fest, dass der Konnektor nur auf bestimmt HTTP-Operationen reagiert.

Durch die Definition des Ressourcen-Typs haben Sie bereits automatisch zum jeweiligen Ressourcen-Typ passende HTTP-Methoden voreingestellt.

- **GET** (Daten der Ressource abrufen)
- **POST** (neue Ressource anlegen)
- **PUT** (Ressource aktualisieren)
- **DELETE** (Ressource löschen)

→ Siehe *HTTP-Methoden verwenden (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 28.3, S. 290).*

Aufbereitung empfangener Daten

Legen Sie die Kodierung für Daten fest, die als Request im Format `multipart/form-data` eingehen.

28.8.2 Dialog „Konfiguration der Antwort“

(Input Listener Connector)

In diesem Dialog legen Sie die Einstellungen für die zuvor ausgewählten und von Ihrem Konnektor unterstützten HTTP-Methoden fest, die für eine Antwort genutzt werden.

Einstellungen für GET-Anfragen

Legen Sie hier fest, wie die Repräsentation der Antwort-Daten bei GET-Requests aussehen soll. Die Standard-Einstellungen sind bereits sinnvoll gewählt und können belassen werden.

- **Medien-Typ**
  Wählen Sie den gewünschten Medien-Typ aus.

Informationen zu MIME-Typen finden Sie unter http://de.wikipedia.org/wiki/Internet_Media_Type und http://de.selfhtml.org/diverses/mimetypen.htm.

- **Zeichensatz**
  Wählen Sie den Zeichensatz der Antwort-Daten aus.

Einstellungen für POST und PUT-Anfragen

- **Workflow asynchron starten**
  Aktivieren Sie diese Option, wenn der Input Listener nach dem Empfang des Request die HTTP-Statusmeldung direkt an den Client zurückliefern soll. In diesem Fall wird die Antwort vor dem Start des Workflows geschickt. Das Ergebnis des Workflows wird nicht zurückgegeben. Falls die Option nicht aktiviert ist, startet der Workflow synchron und die Antwort wird erst an den Client zurückgeschickt, nachdem das letzte Modul des Workflows ausgeführt wurde.
- **Antwort-Daten setzen**
  Aktivieren Sie diese Option, wenn Sie auf eine Anfrage kein Antwort-Daten zurückliefern wollen. Wenn Sie mit POST, PUT oder DELETE eine Ressource manipulieren, muss nach ausgeführter Aktion nicht unbedingt eine Antwort zurückgesendet werden. Ein Antwortcode reicht in diesem Fall.
  - **Medien-Typ**
  - **Zeichensatz**

  → Siehe *Einstellungen für GET-Anfragen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 28, S. 299)*
- **Antwort-Statuscode**
  Wählen Sie aus, was als Antwort-Statuscode empfangen werden soll.

Um im Fehlerfall in der Response anzuzeigen, dass die Anfrage-Daten fehlerhaft sind, kann die Variable `restConnector.responseStatusCode` für den Antwort-Statuscode auf den HTTP-Status-Code 400 (für „Bad Request“) gesetzt werden.

Allgemeine Einstellungen

- **Antwort-Daten automatisch kodieren**
  Aktivieren Sie diese Option, wenn die Antwort-Daten automatisch mit einer dem Server bekannten und vom Client unterstützten Methode kodiert werden sollen.
- **HTTP-Header bearbeiten**
  Der Button „HTTP-Header bearbeiten“ öffnet den Dialog zum Bearbeiten der Header-Definition für die verfügbaren HTTP-Methoden.
  → Siehe *HTTP-Header (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 28, S. 301)*

28.8.3 Dialog „Konfiguration der Anfrage“

(Medium und Output Connector)

In diesem Dialog zur Konfiguration der anzufragenden Ressource legen Sie die aufzurufende URL, die HTTP-Methode und evtl. eine Authentifizierung fest.

Basiskonfiguration

- **Aufzurufend URL**
  Geben Sie die URL der Ressource ein, an den der Connector seinen Request sendet, z. B. Adresse eines HTTP Input Listeners.
  - **SSL**
    Dient zum Konfigurieren der Server- bzw. Client-Authentifizierung und öffnet den *Dialog „SSL-Konfiguration“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1.1.4, S. 24)*.
- **Methode**
  Geben Sie hier die gewünschte HTTP-Methode an, mit der die Aktion, die auf die Ressource angewendet werden soll, definiert wird.
  → Siehe *HTTP-Methoden verwenden (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 28.3, S. 290).*

Authentifizierung verwenden

Markieren Sie diese Option, wenn der Server eine Authentifizierung fordert. Geben Sie dann den Account ein, den der Connector für die Authentifizierung verwenden soll.

- **Verfahren**
  - **Basic:** Es werden Benutzername und Passwort verlangt und unverschlüsselt übermittelt.

Nur zu verwenden, wenn die Verbindung zwischen Client und Server als sicher betrachtet werden kann!

- **Digest**: Verlangt ebenfalls Benutzername und Passwort. Das Passwort wird verschlüsselt übertragen. Für die Verschlüsselung werden mehrere Parameter verwendet, u. a. zufällig erzeugte Werte.

- **Benutzername**: Benutzername für die Authentifizierung.
- **Passwort**: Passwort für die Authentifizierung.
- **Authentifizierung für Zugriff auf Process Engine verwenden (Passwort aus Portalkonfiguration)**: Verwendung des internen Benutzers (inubitISPortalUser@@@intern)

Anfragedaten

(nur für POST und PUT)

Hier legen Sie die gewünschte Repräsentation der Anfragedaten fest. Wählen Sie dazu aus der Liste der Medien-Typen einen Content-Type aus und definieren Sie einen Zeichensatz.

HTTP-Header

Die Angaben im Request-Header informieren den Server z. B. über die Art und Konfiguration des Clients und die vom Client erwarteten bzw. unterstützten Dokumentformate. Hier legen Sie z. B. fest, in welcher Repräsentation man eine Liste mit angefragten Aufträgen erhalten will.

Der Button „Header bearbeiten“ öffnet einen Dialog. In dem Dialog können Sie den Request-Header durch Doppelklick auf die einzelnen Zeilen der Header-Liste definieren.

Die zulässigen Header sind in der HTTP-Spezifikation festgelegt, siehe http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html#sec5.3.

Test

- Button „Versand der Anfrage testen“
  Zum Prüfen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann. Der Button öffnet einen Dialog, in dem die konfigurierte Verbindung getestet werden kann und eine Antwort mit Status-Code, HTTP-Header und Daten angezeigt wird.

29 RFID Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Der RFID (Radio Frequenz Identifikation) Connector kann event-gesteuert einen Technical Workflow auslösen. Das auslösende Event ist der Eingang einer Nachricht, die von einem iPort-Gerät der Firma IDENTEC SOLUTIONS gesendet wird.

Siehe www.identecsolutions.com

→ Siehe auch

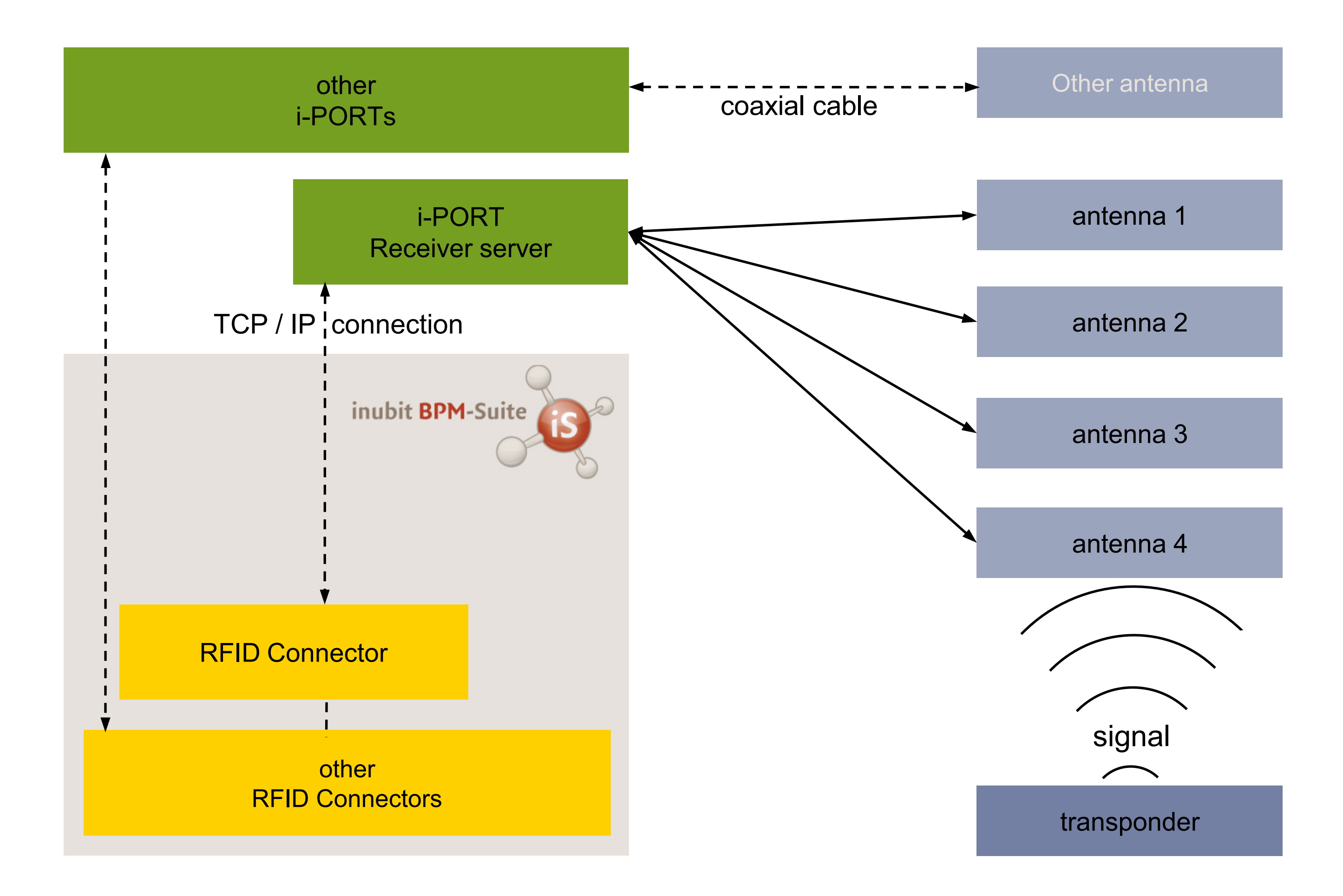
- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

29.1 Funktionsprinzip

Wenn ein Transpondersignal durch die Antennen eines i-Port-Empfängergeräts erfasst wird, schickt dieses eine Nachricht an den RFID Connector und löst damit die Ausführung des Workflows aus, in den der Connector integriert ist.

Ein i-Port kann mit bis zu vier Antennen verbunden sein, die z. B. in verschiedenen Räumen oder Stockwerken eines Gebäudes sein können. Dabei kann jede Antenne alle Transpondersignale empfangen.

Die inubit Process Engine ist über TCP/IP, z. B. über Ethernet, mit dem i-Port verbunden, der RFID Connector fragt über den Port 7070 der inubit Process Engine den i-Port ab und leitet ein empfangenes Signal an den nachgeschalteten Workflow weiter.



29.2 Beispiel-Ausgangsnachricht

Eine Ausgangsnachricht kann z. B. folgendermaßen aussehen:

```
<?xml version="1.0" encoding="UTF-8"?>
<Event>
  <Mask>27</Mask>
  <TagID>223741</TagID>
  <Detection>0</Detection>
  <FieldStrengths_0 antenna="0">
    65
  </FieldStrengths_0>
  <FieldStrengths_1 antenna="1">
    -128
  </FieldStrengths_1>
```

```
  <FieldStrengths_2 antenna="2">
    -128
  </FieldStrengths_2>
  <FieldStrengths_3 antenna="3">
    -128
  </FieldStrengths_3>
  <Time>Wed Apr 07 15:27:36 CEST 2004</Time>
  <IPortID>2</IPortID>
  <maxAntenna>1</maxAntenna>
</Event>
```

Bedeutung der XML Elemente

Die XML-Elemente haben folgende Bedeutungen:

- **TagID**: ID des Transponders, die dieser sendet.
- **FieldStrengths**: Feldstärke, mit der das Signal empfangen wurde. Die Feldstärke wird in Werten zwischen –128 und 128 angegeben. Dabei bedeutet –128 „kein Signal empfangen“.
- **Time**: Datum und Uhrzeit an, zu der das Signal empfangen wurde.
- **IPortID**: Bei der Konfiguration des RFID Connectors geben Sie über den Servernamen und die Portnummer an, von welchem i-Port das Signal empfangen wurde. In diesem Fall hat der i-Port die ID 2.

29.3 Dialog „RFID (IPort) Connector Eigenschaften“

In diesem Dialog haben Sie folgende Optionen:

Konfiguration

- **Servername**
  IP-Nummer oder Servernamen des IPort.
- **Port**
  Standardportnummer ist 7070. Mit dem Button „Standard“ stellen Sie nach Änderungen den Standardwert wieder her.

30 RosettaNet HTTPS Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Der RosettaNet-Standard dient als Basis für den weltweiten Datenaustausch zwischen Firmen mit Schwerpunkt im Bereich Supply Chain Management.

Für mehr Informationen über das RosettaNet Protokoll siehe

- www.rosettanet.org
- „B2Bi am Beispiel von RosettaNet" (Vortrag, gehalten an der Universität Hamburg: http://vsis-www.informatik.uni-hamburg.de/teaching/ss-03/sam/sam03_rn.pdf

Funktionsumfang

Der RosettaNet Connector unterstützt die folgenden Funktionalitäten:

- RosettaNet Implementation Framework (RNIF) V2.0

  RNIF beschreibt den Aufbau der XML-Nachrichten, das Vorgehen beim Ver- und Entpacken der Nachrichten, die unterstützen Protokolle, Sicherheitsanforderungen und die Fehlerbehandlung.
- Asynchrones Empfangen von Nachrichten über HTTP(S)
- Entpacken empfangener Nachrichten, Syntax-Check der Nachrichten und Konvertierung in ein eigenes XML-Format
- Archivierung von Nachrichten
- Erzeugen einer gültigen RosettaNet-Nachricht aus einer im Workflow aufbereiteten PIP-Nachricht

  RosettaNet Partner Interface Processes (PIP) definieren Geschäftsprozesse zwischen Geschäftspartnern.
- Asynchroner Versand von Nachrichten und Empfangsbestätigungen über HTTP(S)
- Erzeugung und Kontrolle der verwendeten Nachrichten-IDs entsprechend dem RNIF

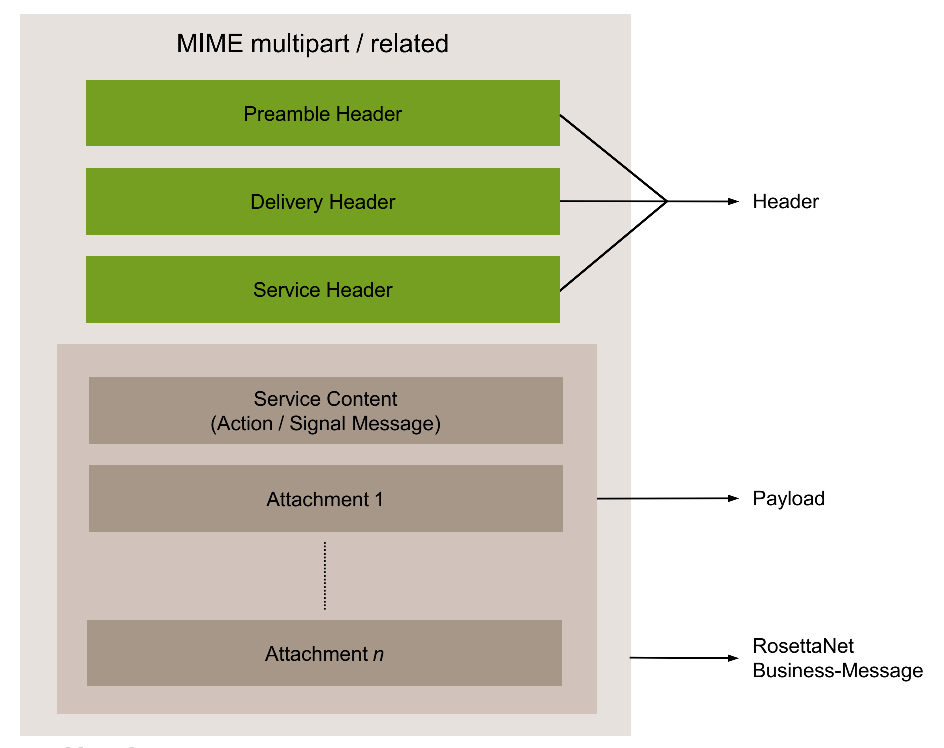
→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

30.1 Struktur und Austausch von RosettaNet-Nachrichten

Struktur

Die folgende Abbildung zeigt die Struktur einer RosettaNet Business Message:



- **Header**
  XML-Metadaten mit Informationen über die Nachricht. Jede Nachricht muss alle Headerteile genau einmal enthalten. Der Header hat folgende Struktur:
  - **Preamble Header**
    Version des RosettaNet Implementation Framework (RNIF), zu dem die Nachricht konform ist. Die aktuelle Version ist RNIF 2.0.
  - **Delivery Header**
    Identifiziert den Absender und den Empfänger. Enthält einen Datums- und Zeitstempel. Alle an der Weiterleitung einer Business Nachricht Beteiligten, also der Absender, der Empfänger und mögliche Zwischenstationen, benutzen diese Informationen für das Routing.
  - **Service Header**
    Informationen über den Partner Interface Process (PIP) als ID, die PIP-Instanz als ID und die Prozessaktivität.
- **Payload**
  Enthält die eigentlichen Nachrichten und ist XML-kodiert.
  - **Service Content**

Gibt an, ob die Nachricht eine „action message“ oder eine „signal message“ ist. Die Prozessaktivität einer „action message“ wird im PIP angegeben (siehe Service Header). „signal messages“ sind Empfangsbestätigungen bzw. Fehlermeldungen.

- **Attachments**

  Wenn die Nachricht eine „action message“ ist, kann diese einen oder mehrere Anhänge enthalten.

Alle Teile einer Business Message sind im MIME/S-MIME-Nachrichtenformat komprimiert.

Bestätigungen

Der Empfang von RosettaNet-Nachrichten wird durch Bestätigungen bzw. Fehlermeldungen signalisiert. Empfangsbestätigungen können synchron oder asynchron versendet werden.

Transportprotokolle

Für den Nachrichtentransport können die Protokolle HTTP und HTTPS benutzt werden.

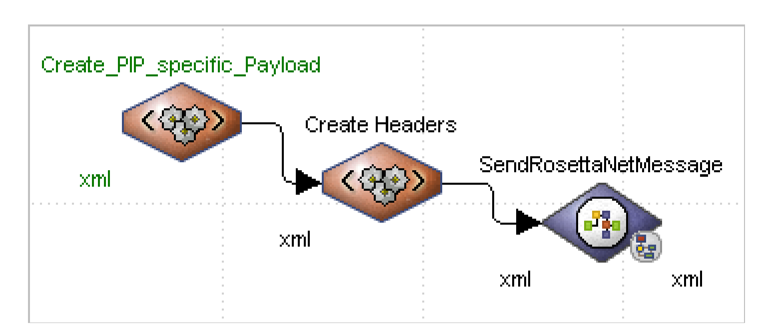
Asynchroner Nachrichtenaustausch

Grundsätzlich unterstützt das PIP-Modell den asynchronen Nachrichtenaustausch, der wie folgt abläuft:

1. Geschäftspartner A schickt eine RosettaNet-Nachricht an Geschäftspartner B. Diese Nachricht enthält eine sog. PIP-ID, über welche die Nachricht eindeutig identifiziert werden kann.
2. B speichert eine Kopie der Nachricht, um den Erhalt nachweisen zu können.
3. B sendet eine Bestätigung an A.
4. Nun gibt es zwei Möglichkeiten:
   - A erhält die Bestätigung.
   - A erhält keine Bestätigung innerhalb des vereinbarten Zeitraums. Dann sendet A die Nachricht noch einmal:

     B empfängt die Nachricht. B überprüft, ob die Nachricht vorher bereits empfangen wurde, weil B sicher stellen muss, dass Nachrichten exakt einmal verarbeitet werden. Dazu wird die PIP-ID der eingegangenen Nachricht mit allen PIP-IDs vorher eingegangener Nachrichten verglichen.

30.2 Beispiel-Workflow: RosettaNet-Nachrichten erzeugen

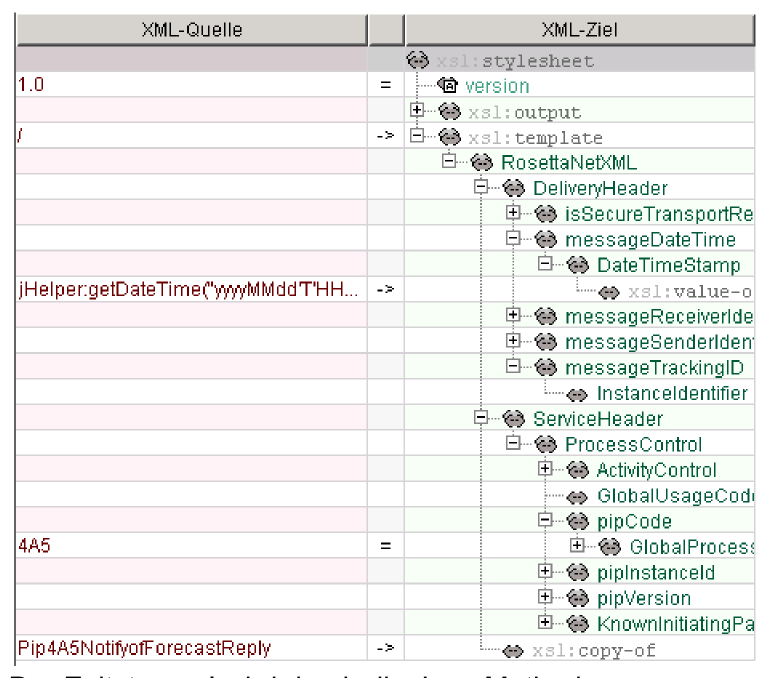


Der abgebildete Workflow erzeugt eine RosettaNet-Nachricht und übergibt diese mit Hilfe des Workflow Connectors „SendRosettaNetMessage" zum Versenden an den nächsten Workflow.

→ Siehe *Beispiel-Workflow: RosettaNet-Nachrichten versenden (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 30.3, S. 311).*

Die Daten für die RosettaNet-Nachricht werden aus dem unternehmenseigenen ERP-System geladen und mit Hilfe von XSLT Convertern in die RosettaNet-Struktur überführt:

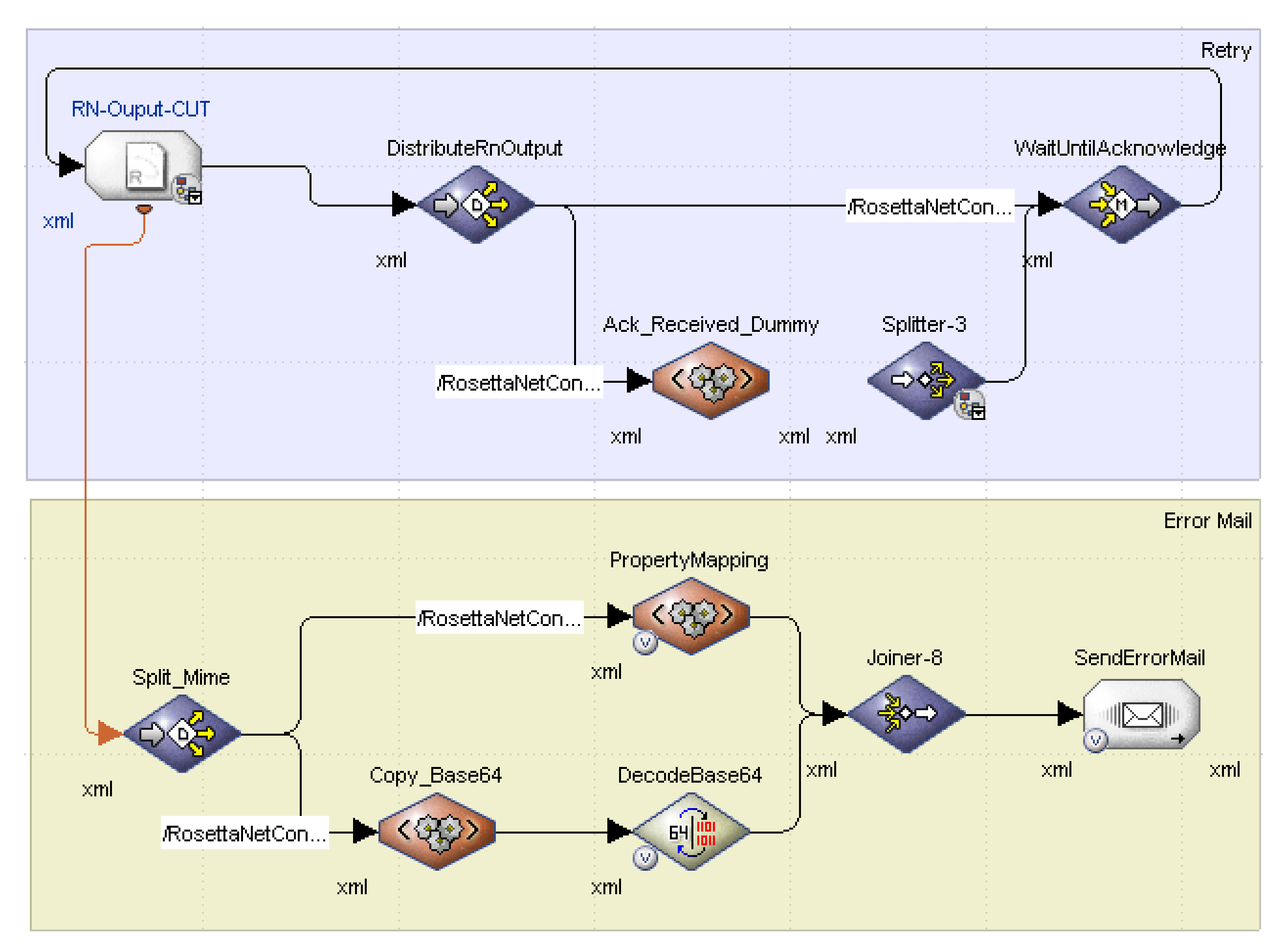
1. Der XSLT Converter „Create PIP-specific Payload" erstellt die PIP-spezifische Nachricht (Payload).
2. Der XSLT Converter „Create Headers" erzeugt die Header. Die folgende Abbildung zeigt ein beispielhaftes Mapping:



Der Zeitstempel wird durch die Java Methode `jHelper:getDateTime(„yyyyMMdd'T'HHmmss.SSS'Z'")` gesetzt. Dem Element `GlobalProcessIndicatorCode` wird die

PIP-ID zugewiesen, die für diese Nachricht vorgeschrieben ist (hier: 4A5). In der letzten Zeile wird der Payload durch einfaches Kopieren hinzugefügt.

30.3 Beispiel-Workflow: RosettaNet-Nachrichten versenden



Der abgebildete Workflow versendet die RosettaNet-Nachricht, die im Abschnitt *Beispiel-Workflow: RosettaNet-Nachrichten versenden (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 30.3, S. 311)* erstellt wurde, und empfängt die Bestätigung über den Eingang der Nachricht. Wenn die Business Message das letzte Mal gesendet wurde und immer noch keine Bestätigung für die Nachricht eingegangen ist, dann wird der Fehlerausgang angesteuert.

Nachricht versenden

Der Output Connector „RN-Output-CUT“ erhält eine RosettaNet Business Message als Eingangsnachricht und verarbeitet diese wie folgt:

1. Der Präambel Header wird hinzugefügt. Er enthält die RNIF-Version.
2. Der Delivery Header und der Service Header wurden im XSLT Converter erstellt, der RosettaNet Output Connector fügt diesen Headern die noch fehlenden Informationen hinzu. Die fehlenden Infos werden den Konfigurationsdaten übernommen, die beim Anlegen des Connectors angegeben wurden. Zusätzlich wird der Instanz-Identifier erzeugt und hinzugefügt.
3. Die für jede Nachricht generierte ID wird in der Indexdatei `ListofGeneratedIds.xml` im Arbeitsverzeichnis gespeichert.

Danach sendet der Output Connector die Nachricht an den Empfänger und hält sie solange im Workflow, bis die Bestätigung eingegangen ist.

Fehler-E-Mail versenden

Der Fehlerausgang wird angesteuert, wenn die Business Message das letzte Mal gesendet wurde und immer noch keine Bestätigung für die Nachricht eingegangen ist. Dann wird über das Mail Connector Modul eine Nachricht gesendet, welche darüber informiert, dass das Versenden der Business Message fehlgeschlagen ist.

Demultiplexer „DistributeRNOutput“

Zusätzlich wird die Nachricht dem Demultiplexer übergeben, der die Nachricht entweder an ein Dummy-Modul oder an den Multiplexer „WaitUntilAcknowlegde“ weiterleitet:

- Wenn der XPath `/RosettaNetConnector/AcknowledgeReceived` nicht leer ist, dann erhält das Dummy-Modul die Eingangsnachricht. Das bedeutet, dass die Nachrichtenbestätigung am Output Connector angekommen und keine weitere Aktion notwendig ist.
- Wenn der XPath `/RosettaNetConnector/Retry` leer ist, dann erhält der Multiplexer „WaitUntilAcknowledge“ die Eingangsnachricht. Dies bedeutet, dass die Bestätigung zu dieser Nachricht noch nicht empfangen wurde.

Splitter Modul

Das Splitter Modul empfängt eine Bestätigung, die vom Input Connector Workflow weitergeleitet wurde, und gibt die Bestätigung an einen Mail-Connector und an den nachfolgenden Demultiplexer weiter.

→ Siehe *Beispiel-Workflow: Nachrichten empfangen und Bestätigung senden (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 30.4, S. 313).*

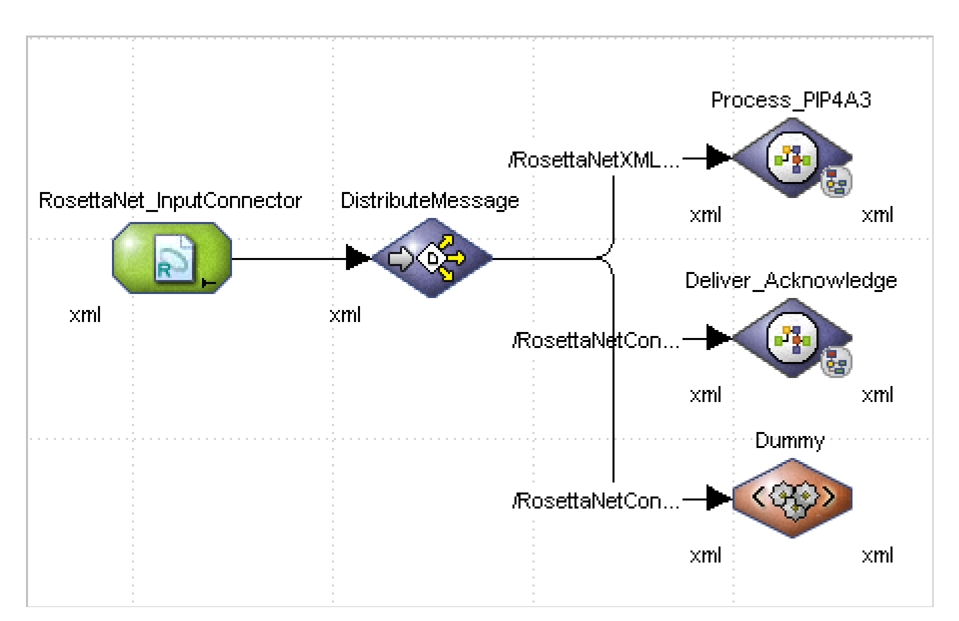
Multiplexer WaitUntilAcknowledge

Der Multiplexer erhält Nachrichten aus dem Demultiplexer „DistributeRNOutput“ sowie Bestätigungsnachrichten aus dem Splitter und wertet beide aus.

Dazu werden die Instanz-ID und die PIP-Nummer der beiden Nachrichteneingänge verglichen. Wenn diese identisch sind, wird eine Nachricht an den Output Connector weitergeleitet, mit der eine vorher verschickte Business Message bestätigt wird. Damit muss diese Business Message nicht mehr im Output Connector vorgehalten werden.

Der Workflow „Send RosettaNet Message“ ist damit beendet.

30.4 Beispiel-Workflow: Nachrichten empfangen und Bestätigung senden



Der abgebildete Workflow empfängt RosettaNet-Nachrichten und leitet diese abhängig von ihrem Inhalt weiter.

Nachrichteneingang

Der RosettaNet Input Connector am Anfang des Workflows ist als Listener konfiguriert und wartet auf Nachrichten. Wenn eine RosettaNet Business Message von einem Geschäftspartner gesendet wird, dann führt der Connector folgende Schritte aus:

- Speichert die Nachricht im angegebenen Arbeitsverzeichnis,
- prüft anhand der `/<PIP>/*.msg`-Dateien im Arbeitsverzeichnis des Moduls, ob die eingegangene Nachricht schon einmal gesendet wurde (wenn ja, dann erzeugt der Connector eine entsprechende XML-Nachricht),

- sendet sofort (synchron) eine Empfangsbestätigung,
- leitet die Nachricht an das Demultiplexer Modul weiter.

Empfangsbestätigungen können auch asynchron versendet werden, z. B. nachdem das wartende Backend-System die RosettaNet-Nachricht erhalten hat. Dazu aktivieren Sie im Input Connector im *Dialog „Asynchrone Empfangsbestätigung“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 30.5.2, S. 320)* den Versand asynchroner Bestätigungen und fügen einen Output Connector in den Workflow ein. Der Output Connector versendet die Bestätigung und sollte daher an einer Position im Workflow eingefügt werden, an der das wartende Backend-System den Empfang der RosettaNet-Nachricht bestätigt hat.

Bedingte Weitergabe

Der nachfolgende Demultiplexer prüft den Inhalt der Nachricht und leitet diese an eines der nachfolgenden Module weiter:

- **Process_PIP4A3**

  Wenn im Service Header im Element `RosettaNetXML/ServiceHeader/ProcessControl/pipCode/GlobalProcessIndicatorCode` der Code 4A3 vorkommt, dann wird die Nachricht zur weiteren Verarbeitung an den Workflow Connector „Process_PIP_4A3“ weitergeleitet.
- **Deliver_Acknowledge**

  Wenn in der Nachricht der XPath `/RosettaNetConnector/AcknowledgeReceived` existiert, dann handelt es sich bei der Nachricht um eine Bestätigungsnachricht, die zur weiteren Verarbeitung an den Workflow Connector „Deliver_Acknowledge“ weitergeleitet wird.

  Der verbundene Workflow enthält den RosettaNet Output Connector, der die Nachricht gesendet hat. Dieser Output Connector wertet die Bestätigungsnachricht aus, um festzustellen, zu welcher Nachricht die Bestätigung gehört. Mit dieser Bestätigung ist klar, dass die Nachricht nicht erneut gesendet werden muss.
- **Dummy**

  Wenn die Nachricht die Zeichenkette `/RosettaNetConnector/MessageAllreadyProcessed` enthält, dann wird die Nachricht an das Dummy-Modul übergeben und nicht weiter verarbeitet.

  An Stelle des Dummys könnte auch z. B. ein Mail Connector stehen, der einen Administrator von dem erneuten Eintreffen einer bereits gesendeten Nachricht informiert. Wichtig ist, dass eine bereits gesendete Nachricht kein zweites Mal verarbeitet wird.

30.5 Dialogbeschreibungen

Dieser Abschnitt erläutert die folgenden Themen:

Beachten Sie die Rechtschreibung, wenn Sie Werte aus der Spezifikation übernehmen, weil in RosettaNet-Nachrichten bei allen Bezeichnungen und Werten zwischen Groß- und Kleinschreibung unterschieden wird!

30.5.1 Dialog „RosettaNet Connector Eigenschaften“

Dieser Abschnitt erläutert die folgenden Themen:

- **Neu (Button)**

  Ein Klick auf „Neu“ fügt eine weitere Registerkarte „Neuer PIP“ für den neuen PIP ein.

  → Siehe *Register „Neuer PIP“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 30.5.1.3, S. 318).*
- **Löschen (Button)**

  a. Zeigen Sie die Registerkarte an, die Sie löschen möchten.

  b. Klicken Sie auf „Löschen“.

  Eine Warnung wird angezeigt. Nach Bestätigung wird der PIP gelöscht.
- Register „**Allgemein**“

  → Siehe *Register „Allgemein“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 30.5.1.1, S. 316).*
- Register „**NoF**“

(Nur Output Connector)

→ Siehe *Register „NoF“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 30.5.1.2, S. 318).*

- Register „**Neuer PIP**“

  (nach Klick auf „Neu“)

  → Siehe *Register „Neuer PIP“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 30.5.1.3, S. 318).*

30.5.1.1 Register „Allgemein“

In diesem Register werden alle Daten angegeben, die für Business Messages zwischen zwei Geschäftspartnern immer gleich sind.

Arbeitsverzeichnis

- **Pfad**

  Der „Auswählen“-Button öffnet einen Dateiexplorer zum Anzeigen der empfangenen und erzeugten Nachrichten. In diesem Verzeichnis wird auch die Datei gespeichert, in der alle erzeugten „Instance Identifier“ enthalten sind. Ein „Instance Identifier“ ist eine Zeichenkette, mit der jede Nachricht im Kontext zweier Geschäftspartner eindeutig gekennzeichnet wird.

Für den RosettaNet Input und Output Connector müssen Sie dasselbe Arbeitsverzeichnis angeben!

DTD Konfiguration

Mit der Auswahl der DTDs wird der syntaktische Aufbau der Nachrichten-Header bestimmt.

→ Die benötigten DTDs können von den Seiten des RosettaNet Konsortiums heruntergeladen werden (ZIP-Datei des aktuellen RNIF V2.0).

- Präambel DTD: `Preamble_MS_V02_00.dtd`
- Delivery Header DTD: `DeliveryHeader_MS_V02_00.dtd`
- Service Header DTD: `ServiceHeader_MS_V02_00.dtd`
- Receipt Ack. DTD: `AcknowledgmentOfReceipt_MS_V02_00.dtd`
- Exception DTD: `Exception_MS_V02_00.dtd`
- NoF DTD: `0A1_MS_V02_00_FailureNotification.dtd`

Um eine bereits geladene Datei zu aktualisieren, müssen Sie eine Datei mit einem anderen Namen laden. Wenn Sie eine Datei mit demselben Namen laden, dann wird die vorhandene Datei nicht ersetzt!

Identifikation

- **Ihre DUNS**: Geben Sie Ihre eigene DUNS Nummer an. Eine DUNS-Nummer dient der eindeutigen Identifikation von Geschäftspartnern.

Sie erhalten eine DUNS-Nummer bei der Dun & Bradstreet Corporation unter http://www.dnb.com.

- **Partner-DUNS**: Geben Sie die DUNS Nummer Ihres Geschäftspartners an.
- **Location ID**: Die Location-ID ist eine frei wählbare alphanumerische Zeichenkette in einem beliebigen Format. Sie kann z. B. benutzt werden, um eine spezielle Niederlassung des Geschäftspartners zu bezeichnen. Auf die Location-ID einigen sich die Geschäftspartner untereinander.

Nachrichtenarchiv aufräumen

- **Nachrichten löschen (nach Tagen)**
  Die Angabe legt fest, nach wie viel Tagen die IDs der bereits empfangenen RosettaNet-Nachrichten aus dem Dateisystem gelöscht werden.
  Die IDs werden in einer XML-Datei gespeichert. Bei einer sehr großen Anzahl von Nachrichten kann diese Datei so groß werden, dass die Performance beeinträchtig wird. Deswegen werden die IDs regelmäßig gelöscht.

Behandlung von Anhängen

- **Anhänge ins Dateisystem speichern**
  (nur beim Input Connector)
  Wenn markiert, dann werden Attachments nach dem Empfang automatisch in Variablen abgelegt. Für jedes Attachment werden drei Variablen erstellt; wenn mehrere Attachments vorhanden sind, dann werden die Variablen durchnummeriert (X), beginnend bei „null“:
  - `RNAttachment.X`
    Inhalt des Attachments, base64-kodiert und komprimiert
  - `RNAttachmentContent-Type.X`
    MIME-Type des Attachments, z. B. application/xml
  - `RNAttachmentContent-ID.X`
    ID des Attachments; kann genutzt werden, um das Attachment in einem RosettaNet-Dokument zu referenzieren.

Fehlerbehandlung

(nur beim Input Connector)

- **Individuell**
  Wählen Sie diese Option, wenn Sie die Fehlerbehandlung im Workflow selbst durchführen wollen. Bei aktivierter Option wirft der Workflow einen Fehler und Sie können diesen behandeln.

- **Signal (Exception)**
  Bei aktivierter Option sendet der Konnektor eine RosettaNet-Signalnachricht mit dem entsprechenden Fehler und der Workflow läuft erfolgreich durch ohne einen Fehler zu werfen.

30.5.1.2 Register „NoF“

(nur Output Connector)

Zum Erstellen einer „Notification of Failure“ (NoF). Ein NoF ist im Prinzip ein PIP0A1 und wird im Fehlerfall versendet, z.B. bei Timeouts innerhalb des Datenaustauschprozesses oder falls Nachrichten nicht verarbeitet werden können.

Die Feldnamen sprechen für sich.

30.5.1.3 Register „Neuer PIP“

Alle Partner Interface Processes (PIPs) sind von RosettaNet spezifiziert, um zu garantieren, dass Sender und Empfänger genormte Nachrichten austauschen.

Ein PIP besteht aus einem von acht möglichen Clustern, jeder Cluster besteht aus einem oder mehreren Segmenten und jedes Segment enthält ein oder mehrere PIP-Beschreibungen. Für jeden PIP gibt es eine Spezifikation. PIPs können für einen Request oder einen Response ausgelegt sein oder auch beide Aktionen unterstützten. Die Struktur der Aktionen ist in einer DTD bzw. einem XML Schema. beschrieben

PIP-Definitionen stehen als Dokumentationspaket (welches u. a. die DTDs bzw. Schemas enthält) auf den Seiten von RosettaNet zur Verfügung und können von dort heruntergeladen werden.

PIP Einstellungen

- **PIP Bezeichnung**
  Ein PIP- Bezeichner wie z. B. 3A2 setzt sich folgendermaßen zusammen:
  - Zahl (0…7), die das Cluster spezifiziert (3=Order Management)
  - Großbuchstabe (A…D), der das Segment angibt (A=Quote and Order Entry)
  - Maximal zweistellige Zahl (aus dem Intervall 1…20), das den PIP angibt (2=Request Price and Availability)
- **PIP Version**
  Wird verwendet zur Überprüfung, ob die Nachricht gültig ist. Es wird ein reiner Zeichenkettenvergleich durchgeführt.

Es gibt Geschäftspartner, die sich darauf einigen, bei langen Release- oder Versionsangaben die letzen vier Stellen nicht anzugeben. Verwendet einer der Partner die Release- oder Versionsnummer ohne die letzten vier Stellen und der andere mit den letzten vier Stellen, würde die Nachricht als ungültig erkannt.

Die zu Grunde liegende PIP Version steht in der RosettaNet Spezifikation.

- **DTD/Schema**

  Zum Laden der DTD bzw. des XML Schemas, welche die Aktion des PIP beschreibt.

  Geben Sie zusätzlich an, ob Sie eine DTD oder ein XML Schema laden.

  - **Input Connector**: Wählen Sie die „Request“-DTD bzw. das entsprechende Schema aus, damit eingehende Anfragen auf Korrektheit überprüft werden können.
  - **Output Connector**: Mit der DTD/dem Schema wird die Ausgangsnachricht auf Korrektheit überprüft.

Um eine bereits geladene Datei zu aktualisieren, müssen Sie eine Datei mit einem anderen Namen laden. Wenn Sie eine Datei mit demselben Namen laden, dann wird die vorhandene Datei nicht ersetzt!

- **Nachweisbarkeit der Empfangsbestätigung**

  (Nur Input Connector)

  Wenn markiert, dann wird eine Hashsumme über die Originalnachricht erzeugt und mit der Empfangsbestätigung zurück gesendet. Damit ist nachweisbar, dass die Originalnachricht beim Empfänger angekommen ist.

- **Sendewiederholungen**

  - **Input Connector**:

    Die Anzahl der Sendewiederholungen ist für jeden PIP in der Tabelle 3-3 einer PIP Spezifikation unter „Retry Count“ angegeben. Für den PIP 3A2 ist zum Beispiel der Wert „3“ angegeben.

  - **Output Connector**:

    Die Anzahl der Sendewiederholungen ist für jeden PIP in der Tabelle 3-3 einer PIP Spezifikation unter „Retry Count“ angegeben. Für den PIP 3A2 ist der Wert z. B. „3“.

- **Rollename Absender/Rollenname Empfänger**

  (Nur Output Connector)

  Die Rollennamen sind in der Spezifikation des jeweiligen PIPs festgelegt. Üblicherweise sind die Rollen in der Tabelle 3-1 der PIP Spezifikation zu finden.

  Für den PIP 3A2 z. B. ist der Rollenname des Absenders „Customer“ und der des Empfängers „Product Supplier“.

- **Servicename Absender/Servicename Empfänger**:
  (Nur Output Connector)
  Die Bezeichnungen sind von RosettaNet spezifiziert und für jeden PIP festgelegt. Für den PIP 3A1 sind die Werte z. B. in der Tabelle 4-1 zu finden.
- **Business Aktivität**
  Die Bezeichnungen für Business Activities sind von RosettaNet spezifiziert und für jeden PIP festgelegt. Üblicherweise ist die Bezeichnung für eine Business Activity in der Tabelle 3-1 der PIP Spezifikation zu finden. Für den PIP 3A2 ist dies z. B. „Request Price and Availability".
- **Business Aktion**
  Die Bezeichnungen sind von RosettaNet spezifiziert und für jeden PIP festgelegt. Üblicherweise ist die Bezeichnung für eine Business Action in der Tabelle 4-2 der PIP Spezifikation zu finden. Für den PIP 3A2 ist dies z. B. je nach Aktion entweder „Price and Availability Request Action" oder „Price and Availability Response Action".
- **Verwendungszweck**
  (Nur Output Connector)
  Fügt dem Service Header der Nachricht einen Hinweis über die Art der Nachricht hinzu (`<GlobalUsageCode>`Test`</GlobalUsageCode>` oder `<GlobalUsageCode>`Production`</GlobalUsageCode>`.
  RosettaNet spezifiziert nicht, wie ein Geschäftspartner mit diesem Element verfahren muss.

30.5.2 Dialog „Asynchrone Empfangsbestätigung"

(nur Input Connector)

Asynchroner Versand der Empfangsbestätigung

- **Asynchron versenden**
  Aktiviert den Versand asynchroner Empfangsbestätigungen.
  → Siehe *Beispiel-Workflow: Nachrichten empfangen und Bestätigung senden (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 30.4, S. 313).*

30.5.3 Dialog „S/MIME-Konfiguration“

(Input Connector)

In diesem Dialog geben Sie die Informationen an, die nötig sind, um verschlüsselte und/oder signierte Eingangsnachrichten zu entschlüsseln bzw. zu prüfen.

Sie benötigen eine Keystore-Datei, die Ihren privaten Schlüssel und den öffentlichen Schlüssel Ihres Geschäftspartners enthält.

Keystore-Konfiguration

Klicken Sie auf den Button, um Ihren Keystore zu laden.

S/MIME-Entschlüsselung

Zum Aktivieren/Deaktivieren der Entschlüsselung.

- **Alias**
  Zur Auswahl des Alias, unter dem der benötigte private Schlüssel im Keystore gespeichert ist.
- **Passwort**
  Passwort für den gewählten Schlüssel.

S/MIME-Signaturprüfung

Zum Aktivieren/Deaktivieren der Signaturprüfung.

- **Alias:** Zur Auswahl des Alias, unter dem der benötigte öffentliche Schlüssel im Keystore gespeichert ist.

30.5.4 Dialog „Verbindungsdaten für Geschäftsnachrichten“

(Input Connector)

Grundkonfiguration

- **Server-URL**
  URL und Port des Servlets, welches auf RosettaNet-Nachrichten wartet.

Authentifizierung

- **Authentifizierung erforderlich**
  Markieren Sie diese Option, wenn der Server eine Authentifizierung fordert. Geben Sie dann den Account ein, den der Connector für die Authentifizierung verwenden soll.
- **Benutzername**: Benutzername für die Authentifizierung.

- **Passwort**: Passwort für die Authentifizierung.

HTTP Header Konfiguration

Über den Header können Informationen wie z. B. Dateigröße, HTTP-Server- und User-Agent-Kennung oder MIME-Typ zwischen Client und HTTP-Server übertragen werden.

Der Button „Header Liste“ öffnet einen Dialog, in dem Sie Name/Wert-Paare als Header definieren können.

Informationen über zulässige Header finden Sie in der HTTP-Spezifikation, siehe http://www.w3.org/Protocols/HTTP/1.1/rfc2616.pdf.

30.5.5 Dialog „Empfangsbestätigungen“

(Output Connector)

Ein Output Connector zum Versand asynchroner Empfangsbestätigungen kann an einer beliebigen Stelle in dem Workflow platziert werden, der die RosettaNet Nachrichten empfängt.

Die nötigen Daten holt sich der Output Connector aus einer Variablen, die von dem RosettaNet-Input Listener am Anfang des Workflow gefüllt wurde.

→ Siehe *Beispiel-Workflow: Nachrichten empfangen und Bestätigung senden (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 30.4, S. 313).*

Versand der Empfangsbestätigung

Versand der Empfangsbestätigung

Wenn markiert, dann werden asynchrone Empfangsbestätigungen versendet.

30.5.6 Dialog „S/MIME-Konfiguration“

In diesem Dialog geben Sie die Informationen an, die nötig sind, um Ausgangsnachrichten zu verschlüsseln bzw. zu signieren.

Sie benötigen eine Keystore-Datei, die Ihren privaten Schlüssel enthält.

Keystore-Konfiguration

Klicken Sie auf den Button, um Ihren Keystore zu laden.

S/MIME-Verschlüsselung

Zum Aktivieren/Deaktivieren der Verschlüsselung.

- **Alias**
  Zur Auswahl des Alias, unter dem der private Schlüssel, der zum Verschlüsseln verwendet wird, im Keystore gespeichert ist.
- **Verschlüsselungsverfahren**
  Zur Auswahl des Verschlüsselungsverfahrens.
- **Payload Container verschlüsseln/Payload verschlüsseln**
  → Siehe RNIF Core Specification, Seite 43 ff.

S/MIME Signierung

Zum Aktivieren/Deaktivieren der Signierung.

- **Alias**
  Zur Auswahl des Alias, unter dem der private Schlüssel, der zum Signieren verwendet wird, im Keystore gespeichert ist.
- **Passwort**
  Passwort für den privaten Schlüssel.
- **MIC Verfahren**
  Zur Auswahl des Message Integrity Check-Verfahrens. Mit diesem Verfahren wird eine Prüfsumme über die Ausgangsnachricht erstellt. Anhand dieser Prüfsumme kann der Empfänger prüfen, ob die Ausgangsnachricht unverändert eingetroffen ist.

30.5.7 Dialog „Verbindungsdaten für Empfangsbestätigungen“

(Input Connector)

Grundkonfiguration

- **Server-URL**
  URL und Port des Servlets, welches auf RosettaNet-Nachrichten wartet.
- **SSL**
  Der Button öffnet den *Dialog „SSL-Konfiguration“ (Workbench/ Process Engine: Systemkonnektor-Guide, Kap. 1.1.4, S. 24)*.

Authentifizierung

- **Authentifizierung erforderlich**

Markieren Sie diese Option, wenn der Server eine Authentifizierung fordert. Geben Sie dann den Account ein, den der Connector für die Authentifizierung verwenden soll.

- **Benutzername**: Benutzername für die Authentifizierung.
- **Passwort**: Passwort für die Authentifizierung.

HTTP Header Konfiguration

Über den Header können Informationen wie z. B. Dateigröße, HTTP-Server- und User-Agent-Kennung oder MIME-Typ zwischen Client und HTTP-Server übertragen werden.

Der Button „Header Liste“ öffnet einen Dialog, in dem Sie Name/Wert-Paare als Header definieren können.

Informationen über zulässige Header finden Sie in der HTTP-Spezifikation, siehe http://www.w3.org/Protocols/HTTP/1.1/rfc2616.pdf.

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

30.5.8 Dialog „RosettaNet Verbindungsdaten“

(Output Connector)

Grundkonfiguration

- **Server-URL**
  URL des Servers, mit dem sich der Konnektor verbinden soll.
- **SSL**
  Dient zum Konfigurieren der Server- bzw. Client-Authentifizierung und öffnet den *Dialog „SSL-Konfiguration“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1.1.4, S. 24)*.

Authentifizierung

- **Authentifizierung erforderlich**
  Markieren Sie diese Option, wenn der Server eine Authentifizierung fordert. Geben Sie dann den Account ein, den der Connector für die Authentifizierung verwenden soll.
- **Benutzername**: Benutzername für die Authentifizierung.

- **Passwort**: Passwort für die Authentifizierung.

HTTP Header Konfiguration

Über den Header können Informationen wie z. B. Dateigröße, HTTP-Server- und User-Agent-Kennung oder MIME-Typ zwischen Client und HTTP-Server übertragen werden.

Der Button „Header Liste“ öffnet einen Dialog, in dem Sie Name/Wert-Paare als Header definieren können.

Informationen über zulässige Header finden Sie in der HTTP-Spezifikation, siehe http://www.w3.org/Protocols/HTTP/1.1/rfc2616.pdf.

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

31 SAP Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Der SAP Connector verbindet sich zu einem SAP-System und bietet folgende Kommunikationsarten mit dem SAP-System:

- **BAPIs/RFCs**

  Der SAP Connector kann BAPIs und RFCs im SAP aufrufen bzw. von BAPIs/RFCs aufgerufen werden.
- **IDocs**

  IDocs sind ASCII-Dokumente in einem XML-Format mit zusätzlichen Statusinformationen. Der SAP Connector nutzt IDocs, um Daten mit dem SAP System auszutauschen.

Voraussetzungen

- Um die Kommunikation zwischen der inubit Process Engine und dem SAP-System zu ermöglichen, müssen Sie den SAP Java Connector (JCo) installieren.

  → Siehe *SAP Java Connector (JCo) installieren (Workbench/ Process Engine: Systemkonnektor-Guide, Kap. 31.1, S. 328).*
- Der Benutzer, mit dem sich der SAP Connector via JCo mit dem SAP-System verbindet, muss auf dem SAP-System erstellt und eingerichtet sein.
- Die Gateways zur Verbindung mit dem SAP-System sind als Netzwerk-Services in der Datei „/etc/services“ (Linux) bzw. „C:\Windows\System32\drivers\etc\services“ (Windows) eingetragen. Je nach Gateway-Bezeichnung müssen die Einträge wie folgt heißen:
  - `sapgw00 3300/tcp`
  - `sapgw01 3301/tcp`

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

31.1 SAP Java Connector (JCo) installieren

Voraussetzungen

- Sie benötigen einen Zugang für das SAP OSS oder für SAPNet, um die Installationsdateien herunterzuladen.
- In der Installationsbeschreibung von SAP wird darauf hingewiesen, dass ein JDK 1.3 oder höher benötigt wird. Bei der Installation der inubit Suite 6 wurde bereits ein passendes JRE installiert.

So gehen Sie vor

1. Öffnen Sie einen Browser und laden Sie URL http://service.sap.com/connectors.
2. Wählen Sie „SAP Java Connector > Download“.
3. Laden Sie folgende Bibliotheken herunter und entpacken Sie diese:
   - **SAP IDOC Class Library**, z. B. `sapidoc3_3.0.4.zip`
     Enthält `sapidoc3.jar`.
   - **SAP Java Connector** (passend zu Ihrem Betriebssystem), z. B. `sapjco3-NTintel-3.0.5.zip`
     Enthält
     - `sapjco3.jar`
     - abhängig von Ihrer Auswahl z. B. `sapjco3.dll` (Windows) oder `libsapjco3.so` (Linux)
4. Laden Sie die beiden `*.jar`-Dateien als Treiber in die inubit Process Engine hoch.
   → Siehe *Treiber installieren (Process Engine: Administrator- und Entwickler-Guide, Kap. 3.4, S. 51).*

Falls Sie eine Fehlermeldung erhalten, die auf das Fehlen der Middleware-Schicht hinweist, lesen Sie die SAP-Note 684106 (zu finden im SAP Marketplace) und wenden Sie die dort beschriebene Lösung an.

5. Abhängig von Ihrem Betriebssystem kopieren Sie die `*.dll/so`-Dateien in eines der folgenden Verzeichnisse:
   - **Windows**:
     - Bei Start der inubit Process Engine über das Startmenü: `<iS-installdir>\_jvm\jre\bin`
     - Bei Einsatz der inubit Process Engine als Dienst unter Tomcat: `<iS-installdir>server\Tomcat\bin`
   - **Linux 32-Bit**
     `<iS-installdir>/_jvm/jre/bin`
   - **Linux 64-Bit**
     `<iS-installdir>/_jvm/jre/lib/amd64`

Unter Umständen müssen Sie zusätzlich das Verzeichnis in die Umgebungsvariable PATH aufnehmen.

6. Starten Sie die inubit Process Engine neu.

31.2 Funktionsprinzip

Typ	Modus	Funktion
Input Connector (Listener)	BAPI/RFC synchron	Der SAP Connector wird von einer Funktion im SAP-System aufgerufen. Die Funktion wird vom SAP Connector und den nachfolgenden Modulen ausgeführt. Danach werden die Rückgabewerte in Form einer BAPI-konformen XML-Response an die aufrufende Funktion zurückgegeben. Zum Erstellen der XML-Reponse wird ein XSLT Converter Modul und darin der SAP Explorer verwendet.
	BAPI/RFC asynchron	Der SAP Connector wird von einer Funktion im SAP-System aufgerufen. Die Funktion wird vom SAP Connector und den nachfolgenden Modulen ausgeführt, evtl. vorhandene Rückgabewerte werden nicht an das SAP zurückgegeben.
	IDoc	Der SAP Connector wird vom SAP System aufgerufen und erhält von diesem ein IDoc; das IDoc wird von nachfolgenden Modulen im Workflow verarbeitet.
Medium/Output Connector	BAPI/RFC	Der SAP Connector ruft eine Funktion auf dem SAP-System auf. Mit einem XSLT Converter Modul und dem SAP-Explorer wird der XML-Request erstellt, den das vom SAP Connector anzusprechende BAPI erwartet. Der XML-Request enthält den Namen der aufzurufenden Funktion sowie Parameter und Werte. Der XSLT Converter übergibt den XML-Request an den Connector, der SAP Connector ruft damit die Funktion auf dem SAP System auf, dann wird die Funktion dort ausgeführt. Das Ergebnis erhält der SAP Connector und gibt es an das nachfolgende Modul weiter.
	IDoc	Der SAP Connector erhält als Eingangsnachricht ein oder mehrere IDocs. Der SAP Connector verbindet sich mit dem SAP System und sendet die IDocs.

31.3 XML-Request/Response erstellen

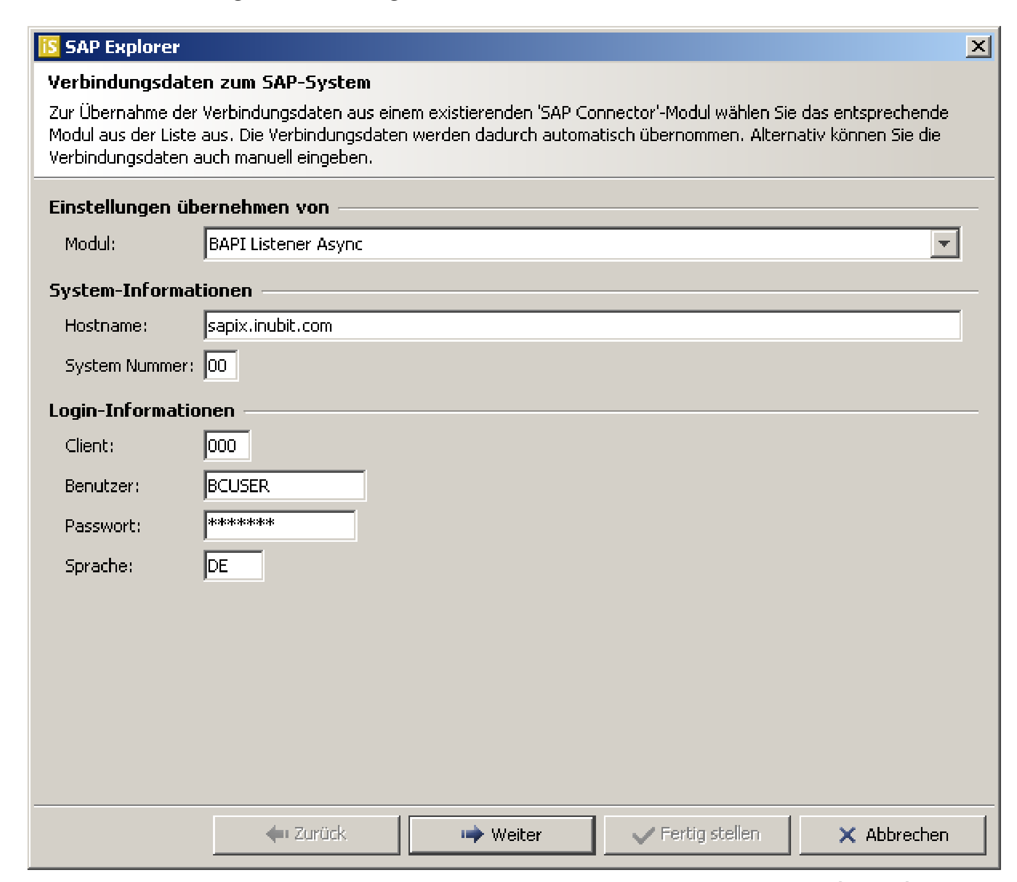
Wenn Sie den SAP Connector mit einer der Kommunikationsarten BAPI/RFC oder IDoc verwenden, benötigen Sie eine konforme XML-Response bzw. einen entsprechenden XML-Request. Zum Erstellen verwenden Sie einen XSLT Converter und den SAP Explorer.

Erzeugen und publizieren Sie zuerst den SAP Connector, damit dessen Verbindungsdaten zum SAP System automatisch

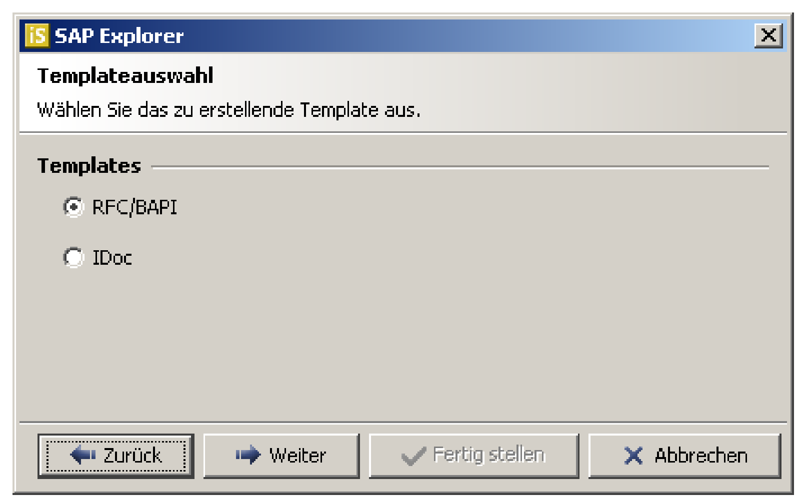
übernommen werden können. Andernfalls müssen Sie die Verbindungsdaten manuell eingeben.

So gehen Sie vor

1. Erzeugen Sie einen XSLT Converter.
2. Öffnen Sie das Modul im lokalen Arbeitsverzeichnis zum Bearbeiten. Das Register „XSLT Converter“ wird angezeigt.
3. Öffnen Sie im Bereich „XML Zieldatei“ das -Menü.
4. Wählen Sie „Öffnen von… > SAP Explorer“:
   Der folgende Dialog öffnet sich:



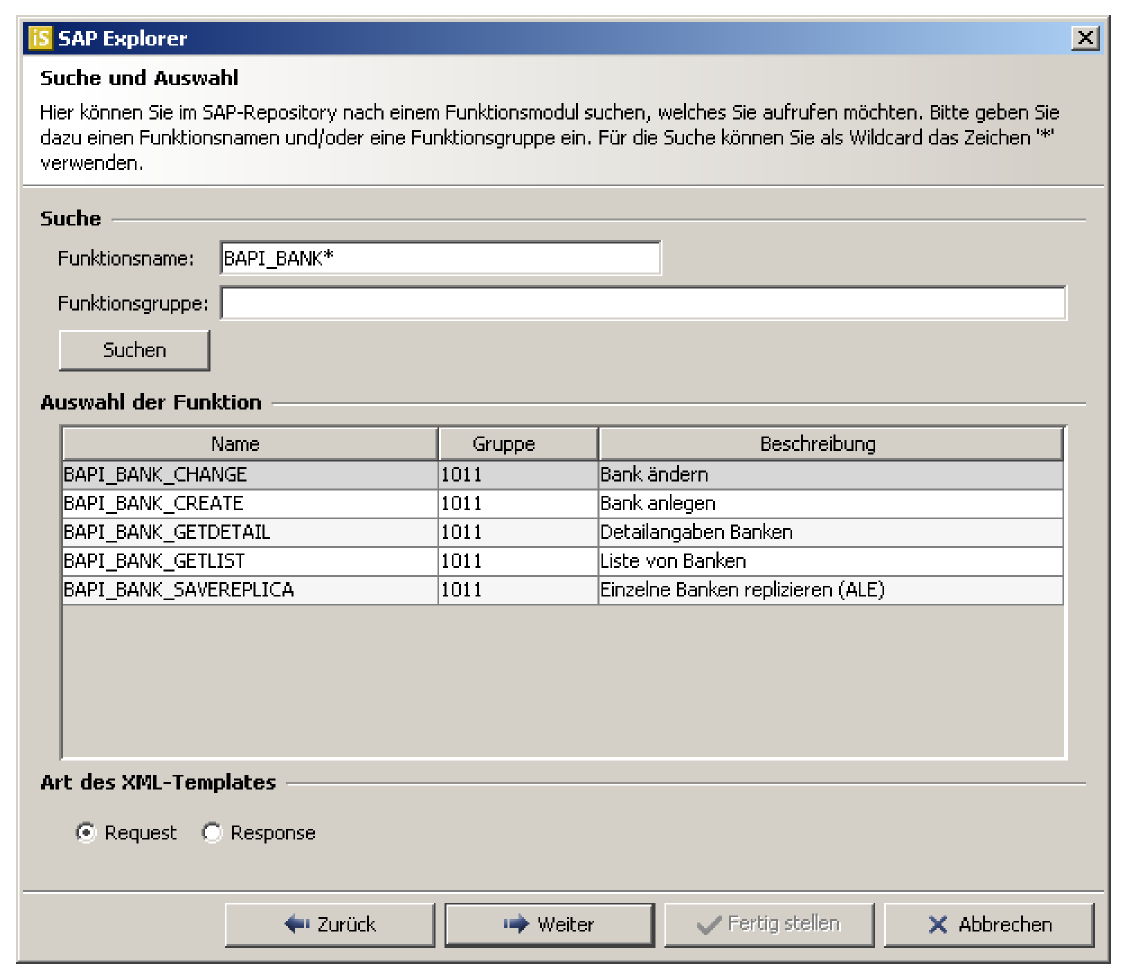
   In diesem Dialog werden die Verbindungsdaten zum SAP System angezeigt.
   Wählen Sie aus der Liste „Modul“ das SAP-Modul mit den passenden Verbindungsdaten oder geben Sie die Verbindungsdaten manuell ein.
5. Klicken Sie auf „Weiter“. Der folgende Dialog wird angezeigt:



Markieren Sie eine der Optionen.

6. Klicken Sie auf „Weiter“. Die folgenden Dialoge sind abhängig von der Art des ausgewählten Templates:

   - **Bei Auswahl „RFC/BAPI“**:



Dieser Dialog unterstützt Sie bei der Auswahl der Funktion, für die Sie einen Request bzw. eine Response benötigen.

Geben Sie den Funktionsnamen oder die Funktionsgruppe bzw. Teile davon ein. Sie können den Asterisk (*) als Wildcard verwenden. Klicken Sie auf „Suchen“, um eine Liste mit passenden Funktionen angezeigt zu bekommen.

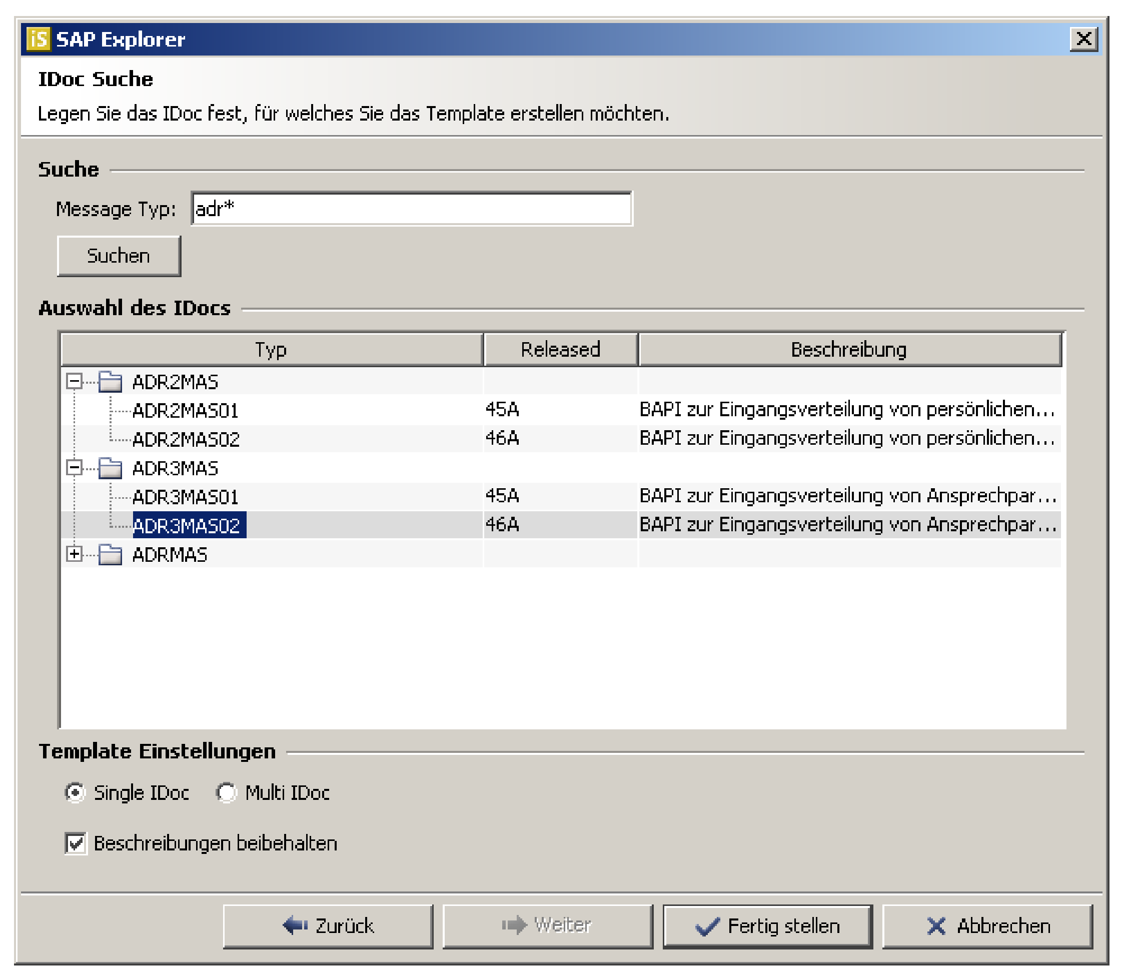
Wenn Ihre Suche ein Ergebnis geliefert hat, markieren Sie in der Tabelle die Funktion, die Sie benutzen möchten.

Dann legen Sie fest, welche Art von XML-Template erzeugt werden soll:

- Response: Wenn Sie den SAP Connector als synchronen Input Listener verwenden
- Request: Beim Einsatz als Medium oder Output Connector

Klicken Sie auf „Weiter“. Fahren Sie fort mit Schritt 7.

- **Bei Auswahl „IDoc“**:



Dieser Dialog unterstützt Sie bei der Auswahl des Nachrichtentyps, für den das Template erstellt wird,

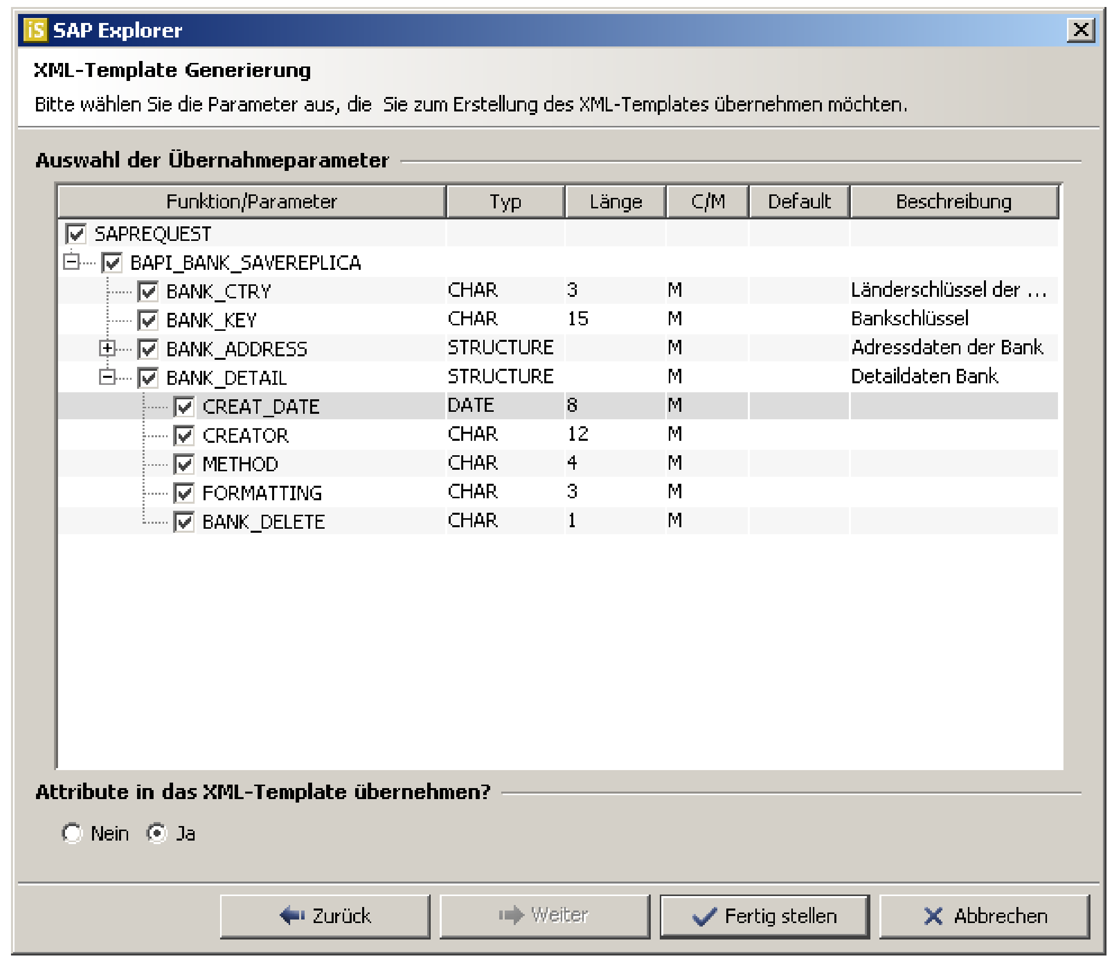
Geben Sie den Namen des IDocs oder Teile davon ein. Sie können den Asterisk (*) als Wildcard verwenden. Klicken Sie auf „Suchen“, um eine Liste mit passenden IDocs angezeigt zu bekommen.

Wenn Ihre Suche ein Ergebnis geliefert hat, markieren Sie in der Tabelle das IDoc, welches Sie benutzen möchten. Legen Sie dann fest, welche Art von XML-Template erzeugt werden soll: Single IDoc oder Multi IDoc.

Klicken Sie auf „Fertigstellen“. Der Dialog schließt sich und das Template wird im Bereich „XML-Ziel“ angezeigt.

Fahren Sie fort mit Schritt 8.

7. Bei Auswahl „BAPI“ öffnet sich der folgende. Dialog:



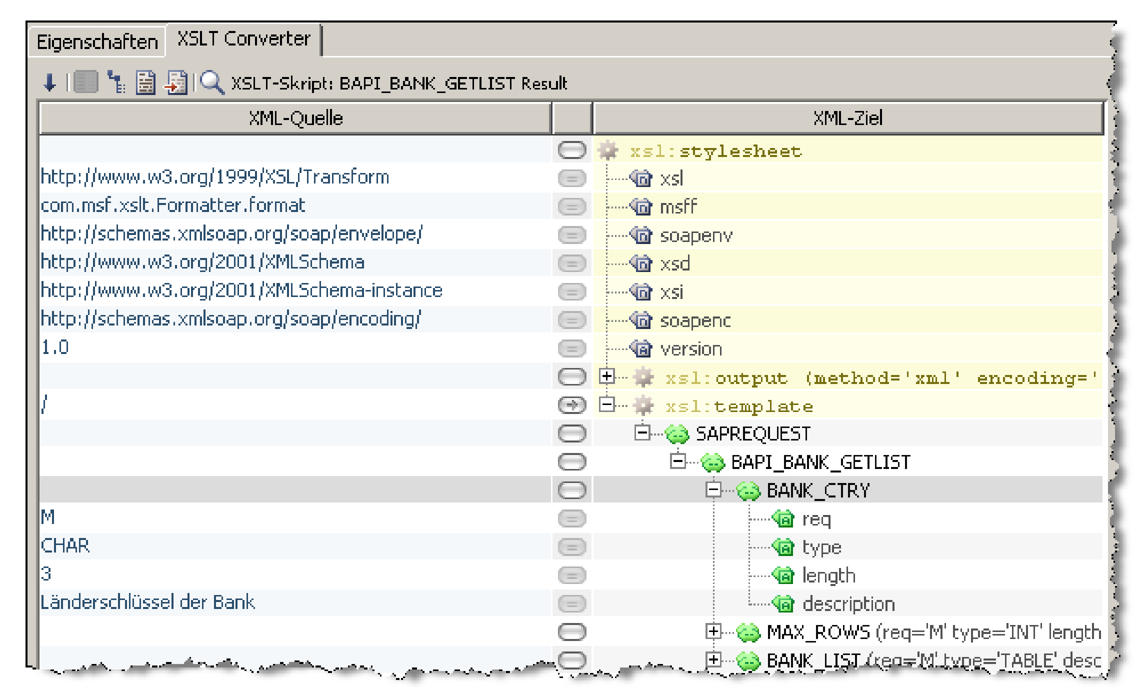
Markieren Sie die Checkboxen der Parameter, die als Elemente in das XML-Template übernommen werden sollen.

Wenn ein Parameter nicht im Template berücksichtigt werden soll, entfernen Sie die Markierung der Checkbox vor dem Parameter.

Geben Sie an, ob die Werte aus den Spalten neben den Funktionen/Parametern als Attribute in das XML-Template übernommen werden sollen.

Klicken Sie auf „Fertig stellen“. Der Dialog schließt sich und das Template wird im Bereich „XML-Ziel“ angezeigt.

8. Für die weitere Bearbeitung des XML-Templates ziehen Sie den Wurzelknoten der Nachricht auf das Element `xsl:template`. Danach sieht das Stylesheet folgendermaßen aus:



9. Um die Parameter mit Werten zu belegen, geben Sie die Werte auf der linken Seite (XML-Quelle) ein. Beachten Sie dabei die Beschränkungen, die durch die Attribute vorgegeben sind (Typ/ Länge).

Damit haben Sie ein Stylesheet erstellt, das eine konforme XML-Response bzw. einen entsprechenden XML-Request erzeugt.

31.4 Dialog „SAP Connector Eigenschaften“

Die Optionen in diesem Dialog sind abhängig vom Konnektortyp und dem Kommunikationsmodus.

Kommunikationsmodus

- **BAPI/RFC**
  Zur Kommunikation mit BAPIs und RFCs. Wenn Sie diese Kommunikationsart in Kombination mit einem Input Listener wählen, dann müssen Sie zusätzlich angeben, ob die Kommunikation synchron oder asynchron stattfinden soll (Option „Modus“).
- **IDoc**

Zur Kommunikation über IDocs.

Der SAP Connector verarbeitet XML IDoc. Wenn Sie EDI-Nachrichten einlesen lassen möchten, müssen Sie diese erst mit einem EDI-XML Adapter nach XML konvertieren. Analog dazu können Sie die XML IDoc-Ausgangsnachrichten des SAP Connectors mit einem XML-EDI Adapter nach EDI konvertieren.

System-Informationen

- **Hostname**: Name des Rechners, auf dem Ihr SAP System läuft.
- **Systemnummer**: Systemnummer (zweistellig).
- **Kodierung** (nur bei Kommunikationsart „BAPI/RFC“): Geben Sie die Zeichenkodierung für die zu kommunizierenden Daten an.

Login Informationen

Ab Version 2.1.6 des SAP Connectors wird bei dem Login (Benutzer/Passwort) Groß- und Kleinschreibung unterschieden!

- **Client**: Dreistellige Clientnummer.
- **Benutzer**: Name des Benutzers, mit dem Sie Zugriff auf das SAP System haben.
- **Passwort**: Zum Benutzernamen passendes Passwort.
- **Sprache**: Sprache, in der die SAP-Systemdialoge angezeigt werden sollen, Eingabe als zweistellige Buchstabenfolge in Großschreibung, z. B. DE, EN oder FR. Angabe ist optional.

Listener-Einstellungen

(Nur bei Input Listener)

Die Gateway-Einstellungen sind notwendig, weil das SAP System nicht direkt, sondern über ein Gateway mit der inubit Process Engine kommuniziert:

- **Gateways Hostname**
  Name des Gateway-Servers.
- **Gateway Servicenummer**
  Da es mehrere Gateway Hosts geben kann, sind diese nummeriert. Fragen Sie Ihren SAP-Administrator nach dieser Nummer, diese kann z. B. so aussehen „sapgw00“.
- **Programm-ID**
  Vor dem Erstellen eines Input Listener Connectors muss im SAP System eine RFC-Destination angelegt werden. Diese hat eine Programm-ID, die hier angegeben werden muss. Fragen Sie Ihren SAP-Administrator nach dieser Programm-ID.
- **Unicodemodus aktivieren**

Markieren Sie diese Option, um mit SAP-Systemen zu kommunizieren, die im Unicode-Modus laufen.

Sonst erhalten Sie bei BAPI/RFC Fehlermeldungen. Bei einem IDoc-Listener kommen IDoc nicht an, obwohl im SAP-System die Meldung hinterlegt ist, dass die Daten erfolgreich an das externe System (die inubit Process Engine) übertragen wurden.

Wenn die inubit Process Engine das Gateway nicht erreichen kann, versucht dieser in 300 Sekunden-Intervallen, die Verbindung wieder herzustellen. Sie können dieses Standardintervall in der SAP Connector-Eigenschaftentabelle ändern (Moduleigenschaft `RetryInterval`, Angabe in Sekunden).

Modus

Synchron/Asynchron

(nur bei Einsatz als Input Listener und BAPI/RFC-Kommunikation):

Legen Sie fest, ob die Rückgabewerte an SAP zurückgegeben werden (synchron) oder nicht (asynchron).

Weitere Einstellungen

Transaktionshandling

(nur bei BAPI/RFC als Medium/Output Connector)

- **Mit Transaktionshandling**: Der SAP Connector setzt explizit ein Rollback bzw. Commit ab.
- **Ohne Transaktionshandling**: Der SAP Connector setzt kein explizites Rollback bzw. Commit ab.

Aufruffehlerbehandlung:

- **Exception werfen**: Bei einem Fehler wird eine Exception geworfen und die Ausführung des Workflows abgebrochen.
- **In den Ausgabestrom schreiben**: Bei einem Fehler wird die Fehlermeldung als XML-Datei ausgegeben und die Ausführung des Workflows fortgesetzt. Die Fehlermeldung kann von einem nachfolgenden Modul behandelt werden.

Multi IDoc Verarbeitung

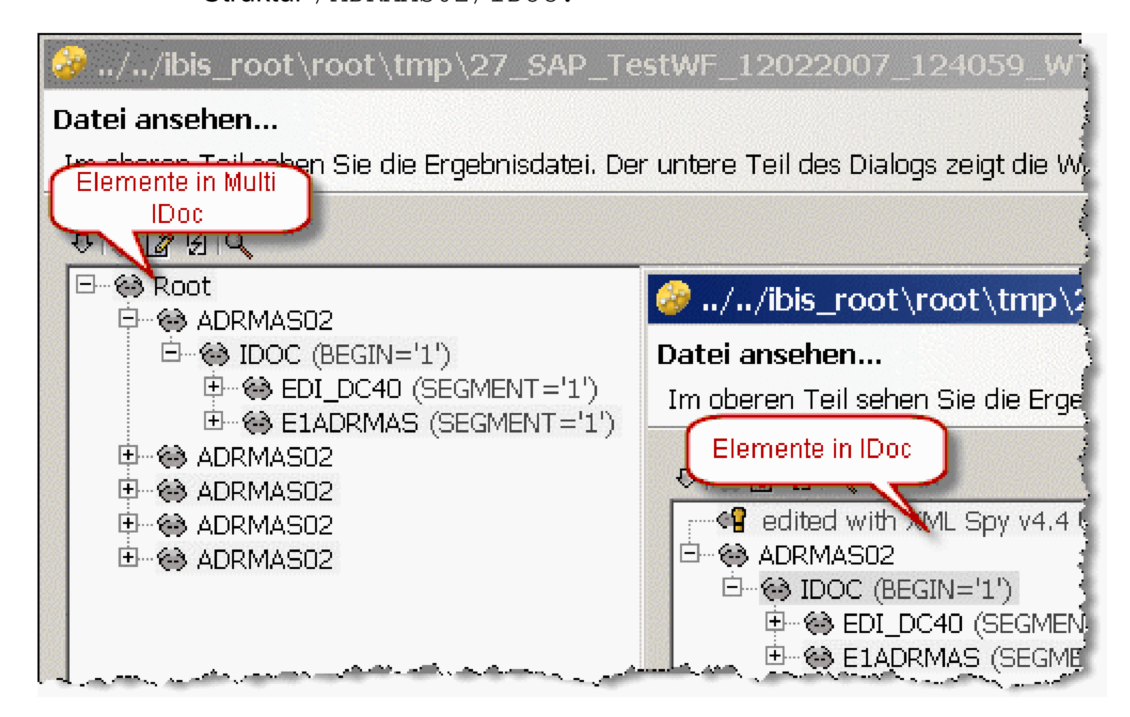
(nur bei Kommunikationsart IDoc)

Aktivieren Sie diese Option, wenn bei der Ausführung eines Workflows mehr als ein IDoc verarbeitet werden soll. Normalerweise wird in einem Workflow genau ein IDoc verarbeitet. Dazu wird über den SAP Connector eine Verbindung zum SAP System aufgebaut, die Nachricht wird verschickt und die Verbindung geschlossen. Wenn man sehr viele IDocs in kurzer Zeit verarbeiten möchte, ist es sinnvoll, die Verbindung zum SAP System aufzubauen und alle Nachrichten auf einmal zu senden.

Die zu sendenden IDocs müssen zu einer Nachricht zusammengefügt sein (Multi IDoc), der Beginn jedes IDoc muss durch ein XML-Element eindeutig gekennzeichnet sein. Das SAP Connector Modul trennt das Multi IDoc am Trennelement und schickt die einzelnen IDocs nacheinander über denselben Verbindungsaufbau.

XML IDoc Trennelement

Geben Sie das XML-Element (inkl. Pfad) ein, welches den Beginn einer IDoc-Nachricht signalisiert. Die Trennung erfolgt inkl. Trennelement, d. h. aus einer XML-Struktur `/Root/ADRMAS02/IDOC` und dem Trennelement `/Root/ADRMAS02` wird ein IDoc mit der Struktur `/ADRMAS02/IDOC`.



Connection Pool

(nur bei Verwendung als Medium/Output Connector, unabhängig von Kommunikationsart)

- **Max. Anzahl von Verbindungen**
  Max. Anzahl von Verbindungen im Connection Pool. Beachten Sie die OSS Notes 314530 und 316877 zum Einrichten externer Verbindungen im SAP System und/oder SAP Gateway.
- **Max. Verbindungswartezeit**
  Legt die max. Wartezeit auf eine verfügbare Verbindung fest (in Millisekunden). Wenn nach der angegebenen Zeit keine Verbindung verfügbar ist, wird eine JCo-Exception mit dem Key JCO_ERROR_RESSOURCE geworfen. Standard sind 30000 Millisekunden (30 Sekunden).
- **Verbindungs-Timeout**

Dieses Timeout betrifft nur „pooled connections“, Verbindungen, die im Connection Pool freigegeben sind und zur Wiederverwendung offen gehalten werden. Es betrifft keine Verbindungen, die aktuell verwendet werden. Eine „pooled connection“ wird geschlossen, wenn sie nach Ablauf der angegebenen Zeit nicht aktiviert wurde. Angabe in Millisekunden, Standard sind 600000 Millisekunden (10 Min.).

Verbindungstest

- **Verbindung testen**

  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

32 Secrypt Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Ein Secrypt Connector bietet folgende Funktionen:

- Dokumente mit einer elektronischen Signatur versehen
- Elektronische Signaturen von Dokumenten prüfen

Voraussetzung

Sie haben Zugriff auf einen Secrypt-Signatur- und Verifier-Server.

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

32.1 Funktionsprinzip

Dokumente signieren

Der Secrypt Connector kopiert das zu signierende Dokument in ein definiertes Eingangsverzeichnis des Secrypt DigiSeal-Servers. Der Secrypt DigiSeal-Server signiert das Dokument und verschiebt es in ein definiertes Ausgangsverzeichnis. Das Ergebnis der Aktion (Dokument erfolgreich signiert/nicht erfolgreich signiert) wird in eine Workflow-Variable geschrieben.

Signaturen prüfen

Der Secrypt Connector kopiert das signierte Dokument in ein definiertes Eingangsverzeichnis des Secrypt Verifiers. Dieser prüft das Dokument und legt das Prüfprotokoll (geprüftes Dokument, Ergebnis und Signatur) in definierte Ausgangs- und Prüfdokumentationsverzeichnisse.

Der Secrypt Connector löscht bei jeder Ausführung alle selbst erstellten Dateien, die in den Eingangs-, Ausgangs- bzw. Prüfdokumentationsverzeichnissen vorhanden sind.

Falls der DigiSeal-Server bzw. Secrypt Verifier nicht auf demselben Rechner wie die inubit Process Engine laufen, muss der Secrypt Connector als Remote Connector ausgeführt werden.

→ Siehe *Modulvariablen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 32.3, S. 343).*

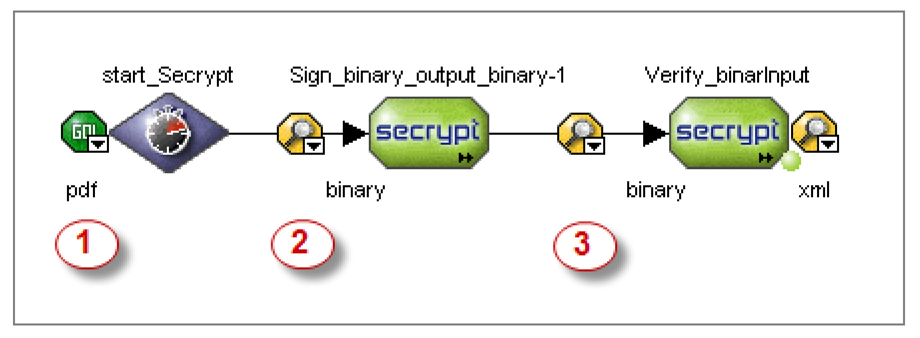
32.2 Beispiel-Workflow

Im folgenden Workflow wird ein PDF-Dokument erst signiert und dann geprüft.

So gehen Sie vor

1. Erstellen Sie einen Technical Workflow mit zwei Secrypt Medium Connectoren.
   - Geben Sie als Nachrichtentyp jeweils „PDF“ an.
   - Wählen Sie beim ersten Connector den Modus „Signieren“ und bei der Eingangs- sowie Ausgangsdatenkonfiguration „Binärdaten“.
   - Wählen Sie beim zweiten Connector den Modus „Prüfen“, bei der Eingangsdatenkonfiguration „Binärdaten“.
     Die Ausgangsdaten (geprüftes Dokument und Prüfergebnisse) werden standardmäßig als XML ausgegeben.
2. Starten Sie den Workflow.

Sie können die Workflow-Zwischenergebnisse an jedem Watchpoint anzeigen, doppelklicken Sie dazu auf den Watchpoint:

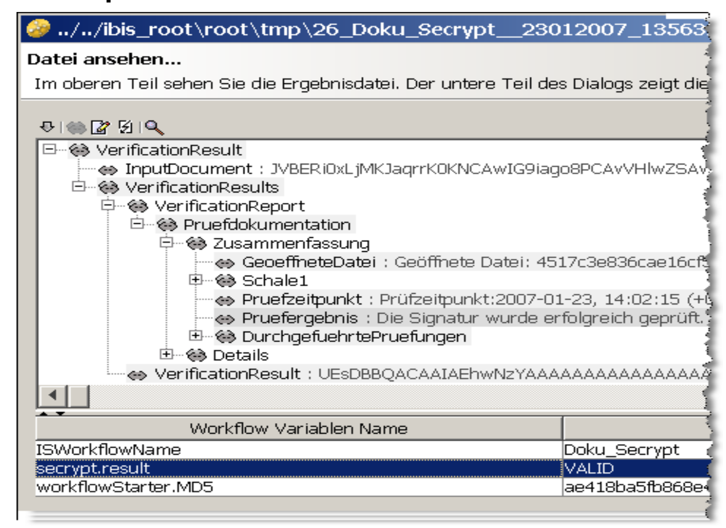


Das Ergebnis:

- **Watchpoint** 1:
  Das Eingangsdokument ist ein PDF-Dokument ohne Signatur.
- **Watchpoint** 2:
  Das Ergebnisdokument ist ein signiertes PDF-Dokument.

Um das PDF-Dokument in einem PDF-Viewer anzusehen, klicken Sie im Dialog „Datei ansehen“ auf „Ergebnisdatei speichern“ und speichern die Datei z. B. als `Test.pdf`. Im PDF-Viewer wird die Signatur im Reiter „Unterschriften“ angezeigt.

- **Watchpoint** 3:



Nach der Prüfung wird das Eingangsdokument in eine XML-Datei eingebettet (Element `InputDocument`). Das Prüfergebnis wird im Element `<Pruefergebnis>` angezeigt.

32.3 Modulvariablen

Bei der Ausführung eines Secrypt Connectors werden die folgenden Modulvariablen gesetzt:

Modulvariable	Inhalt
`secrypt.result`	Prüfergebnis
`secrypt.certificationLevel={advanced, qualified}`	Signaturtyp

32.4 Dialog „Secrypt Connector Eigenschaften“

In diesem Dialog haben Sie folgende Optionen:

Grundkonfiguration

- **Modus**
  Markieren Sie eine der beiden Optionen, um die Funktion des Secrypt Connectors zu definieren:
  - **Signieren**: Das Dokument wird mit einer elektronischen Signatur versehen.
  - **Prüfen**: Die Signaturen des Dokuments werden geprüft.

Pfadkonfiguration des Signaturservers

- **Eingangspfad**
  Pfad zu dem Verzeichnis, in dem der Signaturserver die Dokumente zum Signieren/Prüfen erwartet, z. B. `C:\IS_Repository\Signatur\Verifier_Input`.
- **Ausgangspfad**
  Pfad zum Verzeichnis, in dem der Signaturserver die signierten bzw. geprüften Dokumente bereit stellt. `C:\IS_Repository\Signatur\Verifier_Output`

Die Pfade müssen identisch sein mit den Pfaden, die beim Konfigurieren des DigiSeal Servers/Verifiers angegeben wurden. Die Eingangs- und Ausgangsverzeichnisse dürfen nicht ineinander geschachtelt werden, sondern müssen sich auf derselben Ebene befinden. Eine Angabe für ein Verifier_Output-Verzeichnis wie z. B. `.\Verifier_Input\Verifier_Output` ist also ungültig.

Prüfungsdokumentationsverzeichnisse

(Nur im Modus „Prüfen“)

Der Secrypt Verifier benötigt die folgenden Verzeichnisse, um die Prüfung der Signatur und deren Ergebnis zu dokumentieren:

- **Prüferfolgsverzeichnis**:
  Nach erfolgreicher Prüfung des Dokuments enthält dieses Verzeichnis das geprüfte Dokument, das Prüfergebnis und das Zertifikat.
- **Signaturfehlerverzeichnis**
  Wenn die Prüfung der Signatur nicht erfolgreich war, dann enthält dieses Verzeichnis das geprüfte Dokument und das Prüfergebnis.
- **Systemfehlerverzeichnis**
  Für die Fehlermeldung, wenn beim Prüfprozess Fehler auftraten.

Dokumententyp

- **Typ**
  Geben Sie an, in welchem Datenformat die zu prüfenden bzw. zu signierenden Dokumente dem Secrypt Connector übergeben werden.

Falls Sie im Prüf-Modus als Dokumententyp „p7s“ wählen, wird automatisch XML als Format für die Eingangsnachricht definiert. Der Konnektor erwartet dann eine Eingangsnachricht im XML-Format mit folgender Struktur:

```
<PKCS7>
<Content>
[…Inhalt von ABC.txt, base64-kodiert…]
</Content>
<Signature>
[…Inhalt von ABC.txt.p7s, base64-kodiert…]
</Signature>
</PKCS7>
```

Eingangsdatenkonfiguration

Für die korrekte Verarbeitung müssen Sie angeben, wie die Eingangsnachrichten vorliegen:

- **Binärdaten**
  Zum Einlesen von binären Eingangsdokumenten, z. B. von PDF-Dateien.
- **Base64**
  Zu markieren, wenn die Eingangsdokumente base64-kodiert vorliegen, weil sie z. B. aus XML extrahiert wurden.
- **Signiertes MIME** (nur im Modus „Prüfen“)
  Zu markieren, wenn es sich bei den Eingangsdaten um signierte MIME-Nachrichten gemäß RFC 1847 handelt.
- **XML**
  Zum Einlesen von Eingangsdokumenten, die in XML-Dateien eingebettet sind.
  Geben Sie den XPath zu dem XML-Knoten an, der das Dokument enthält. Alternativ nutzen Sie die Schaltfläche „XML-Knoten auswählen“, um zu diesem Knoten im XML-Dokument zu navigieren.

Ausgangsdatenkonfiguration

(Die folgenden Optionen werden nur beim Signieren von Eingangsdokumenten angezeigt)

Für die korrekte Verarbeitung müssen Sie angeben, wie die signierten Dokumente ausgegeben werden sollen:

- **Binärdaten**
  Gibt das Eingangsdokument binär aus.
- **Base64**
  Gibt die Eingangsdokumente base64-kodiert aus.
- **XML**
  Bettet die Eingangsdokumente in eine XML-Datei ein.

Serverseitige Verzeichnisprüfung

Prüfen

Testet, ob die angegebenen Pfade und Verzeichnisse vorhanden und schreib-/lesbar sind.

33 Security Token Service Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Bei vielen Web Services ist es für den Service-Provider wichtig, die Nutzer der Dienstleistungen zu kennen, z. B. um die Nutzung abrechnen zu können oder um den unerlaubten Zugriff zu verhindern. Dazu können Service-Provider in der Interface-Beschreibung ihres Web Services, in der WSDL-Datei, festlegen, dass sich Service-Consumer gegenüber dem Service-Provider authentifizieren müssen.

Wenn eine größere Zahl von Web Services verwaltet werden soll, ist es sinnvoll, die Authentifizierung auszulagern und eine zentrale Authentifizierungsinstanz, einen so genannten Security Token Service, zu nutzen.

Ein STS Connector fungiert als zentrale Authentifizierungsinstanz für Web Services Consumer und sichert die Kommunikation zwischen Service-Consumer und -Provider zentral auf Nachrichten- und Transportebene.

Konnektortypen

Ein STS Connector wird immer als Input Listener eingesetzt. Er validiert eingehende Authentifizierungs-Requests und sendet bei positivem Validierungsergebnissen Tokens an den aufrufenden Web Service zurück.

Standards

Der STS Connector setzt folgende Web Service-Standards um:

- **WS-Security**
  Standard für die Sicherheit auf Nachrichtenebene. Der Standard legt fest, wie SOAP-Nachrichten signiert und verschlüsselt sowie Sicherheitstoken übertragen werden.

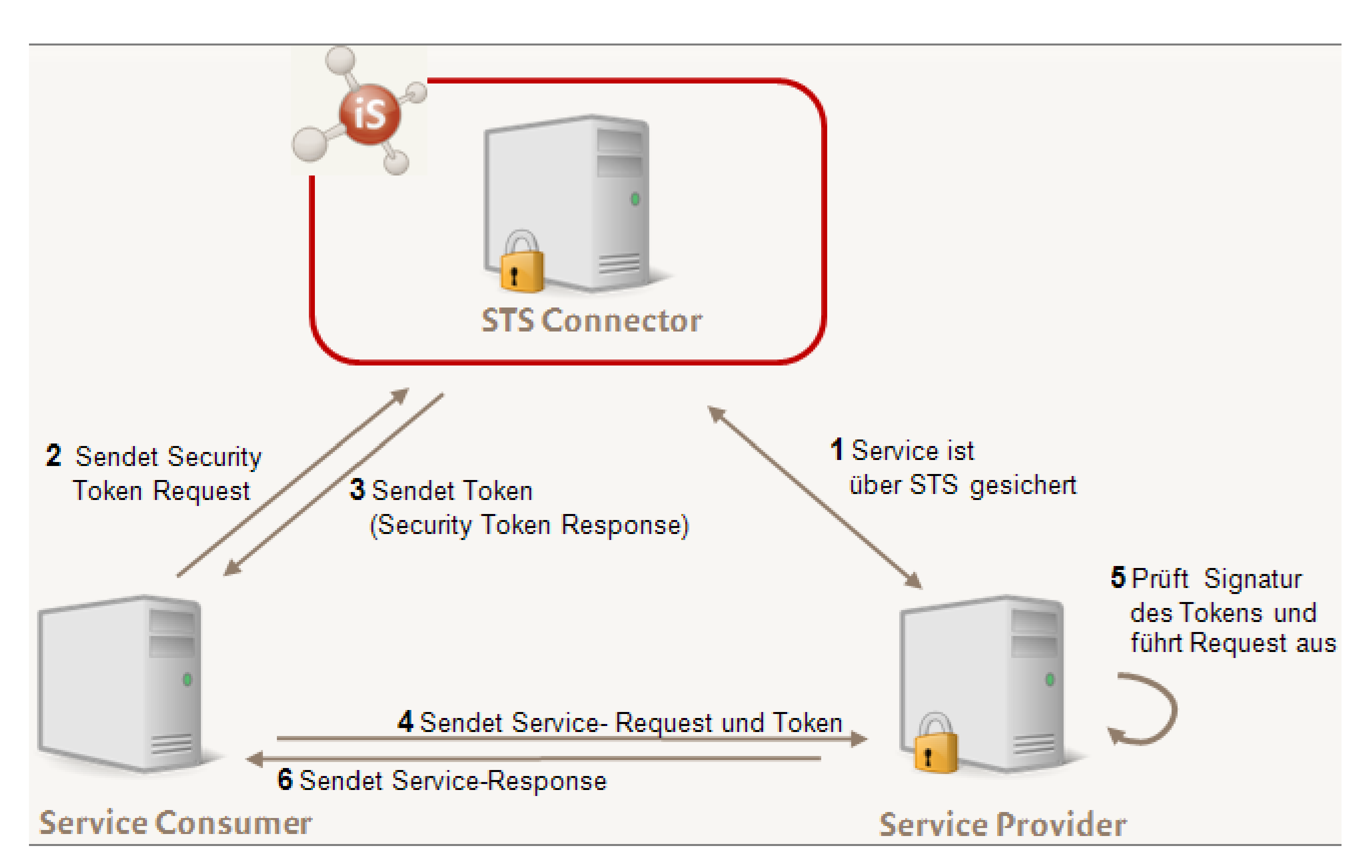
  Siehe http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf
- **WS-Trust**
  Definiert einen speziellen Web Service, den SecurityTokenService, für die Ausgabe, den Austausch und die Validierung von Security Token.

  Siehe http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

33.1 Funktionsprinzip des STS Connectors



Die Absicherung der Kommunikation zwischen Service-Consumer und Provider über einen STS funktioniert folgendermaßen:

1. **Service-Provider bietet Web Service an und nutzt STS**

   Der Service-Provider bietet eine Dienstleistung über einen Web Service an. Der Web Service ist über einen STS gesichert. Ein Web Service-Aufruf ist nur zulässig, wenn der Aufruf ein gültiges Token des STS enthält.

   Wenn der Provider eine Anfrage von einem Consumer erhält, dann wird zuerst die Gültigkeit des Tokens geprüft. Je nach Typ des Tokens prüft der Provider die Gültigkeit über Zertifikate.

2. **Service-Consumer möchte Service nutzen**

   Um den Service aufrufen zu können, benötigt der Consumer dessen Interface-Beschreibung, die so genannte WSDL-Datei. Diese WSDL-Datei kann er direkt von dem Service-Provider oder von einem Verzeichnisdienst wie UDDI anfordern. In der WSDL-Datei sind die Sicherheitsanforderungen beschrieben, die der Consumer erfüllen muss, um den Dienst nutzen zu können. Unter anderem ist angegeben, gegenüber welchem STS sich der Consumer authentifizieren muss.

   Da der Consumer nun die Adresse des STS kennt, kann er diesen aufrufen und sich dort authentifizieren.

3. **STS-Provider prüft Anfrage des Service-Consumers**

   Der STS-Provider prüft die Anfrage des Consumers an Hand der mitgelieferten Merkmale. Diese können Nutzername/ Passwort sein oder das Zertifikat, mit dem die Anfrage signiert ist. Wenn die Prüfung positiv verläuft, dann erzeugt der STS zwei Token:

   - Für das erste Token gilt:
     - Es enthält einen Session Key mit der Information, ob sich der Consumer erfolgreich authentifiziert hat.
     - Das Token ist mit dem öffentlichen Schlüssel des Service-Providers verschlüsselt. Damit ist sichergestellt, dass niemand, außer dem Service-Provider, den Session Key auslesen kann.
     - Das Token ist vom STS signiert, um zu versichern, dass der STS dieses Token ausgestellt hat.
   - Für das zweite Token, das Prüf-Token, gilt:
     - Das Prüf-Token enthält ebenfalls den Session Key. Dieser ist nötig, damit der Consumer seinen Web Service-Aufruf signieren kann.
     - Das Prüf-Token ist mit dem öffentlichen Schlüssel des Consumers verschlüsselt, damit auch der Consumer in den Besitz des Session Keys gelangen kann.

   Der STS sendet Token und Prüf-Token an den Consumer.

4. **Service-Consumer ruft Service-Provider auf**

   Der Consumer empfängt die beiden Token und extrahiert den Session Key aus dem Prüf-Token. Er erstellt den Aufruf des Web Services und signiert den Aufruf mit dem Session Key. Dann sendet der Consumer den Aufruf und das eigentliche Token des STS an den Service-Provider.

5. **Service-Provider prüft die Signatur des Tokens**

   Der Service-Provider prüft die Signatur des Tokens mit dem öffentlichen Schlüssel des STS, um zu verifizieren, dass das Token wirklich vom STS stammt. Anschließend entschlüsselt der Provider das Token mit seinem privaten Schlüssel und erhält damit den Session Key und die darin enthaltenen Aussagen über die Identität des Consumers.

Anhand des Session Keys überprüft der Provider die Signatur des Web Service-Aufrufs. Da zu diesem Zeitpunkt nur der STS, der Consumer und der Service-Provider den Session Key kennen, kann der Service-Provider davon ausgehen, dass wirklich der Consumer, für den der STS das Token ausgestellt hat, die Service-Anfrage signiert hat. Nun muss der Service-Provider nur noch prüfen, ob der STS die benötigten Bescheinigungen in das Token integriert hat.

6. **Service-Provider sendet Service-Response oder Fehlermeldung**

   Wenn die nötigen Bescheinigungen in das Token integriert sind, dann wird der Service-Request ausgeführt, die Service-Response verschlüsselt, signiert und an den Consumer gesendet.

   Wenn die Bescheinigungen fehlen, wird eine Fehlermeldung zurückgegeben.

33.2 Web Services Provider an STS Connector registrieren

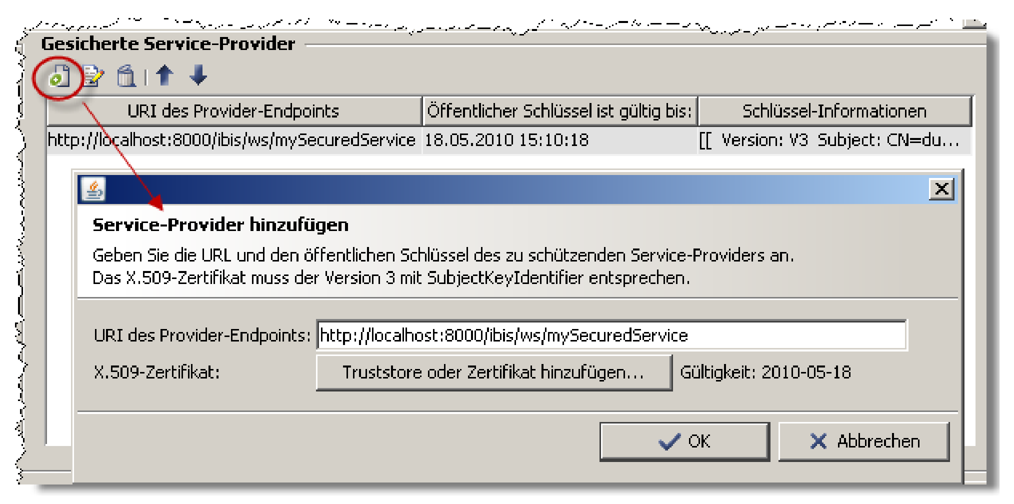
Dieser Abschnitt erläutert, wie Sie beliebige Web Services Provider an einem STS Connector registrieren, um diese abzusichern.

Voraussetzungen

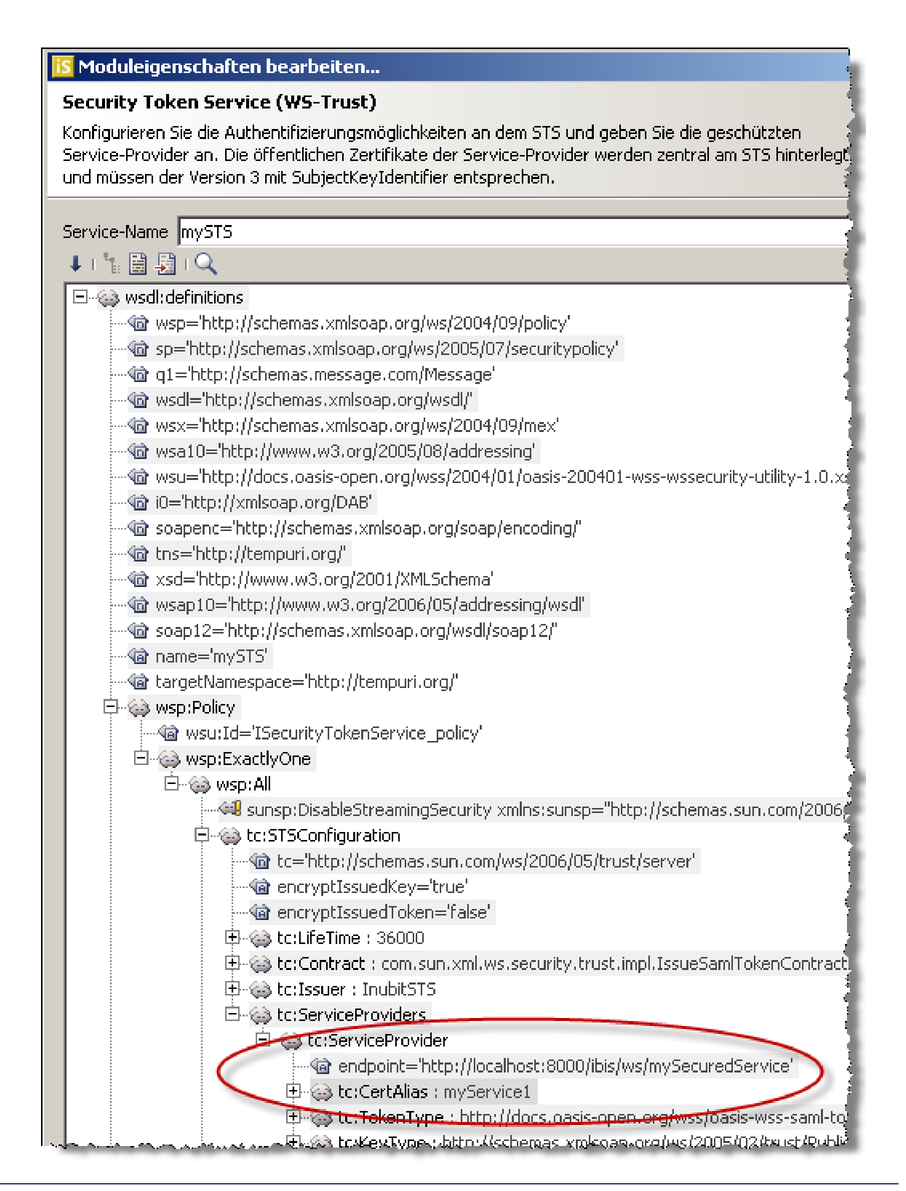
- Die Web Services, die abgesichert werden sollen, sind bereits vorhanden, und die URL, über welche diese Web Services zu erreichen sind, sind bekannt.
- Die Truststores mit den öffentlichen Schlüsseln aller Web Services, die von dem STS gesichert werden sollen, liegen vor.

So gehen Sie vor

1. Erstellen Sie einen STS Connector oder öffnen Sie einen vorhandenen zum Bearbeiten.
2. Klicken Sie im *Dialog „Security Token Service (WS-Trust)“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 33, S. 353)*“ im Bereich „Gesicherte Service-Provider“ auf „Hinzufügen“. Der folgende Dialog öffnet sich:



3. Geben Sie im Feld „URI des Provider-Endpoints“ die URL des Web Service Providers ein.
4. Klicken Sie auf „Truststore oder Zertifikat hinzufügen“, um das Zertifikat des Web Services Providers mit dem öffentlichen Schlüssel hinzuzufügen.
5. Schließen Sie den Dialog mit „OK“.
6. Geben Sie in der WSDL den Alias für das Zertifikat des eben hinzugefügten Web Service Providers an. Die Abbildung zeigt das zu ändernde Element:



Für jeden hinzugefügten Web Service Provider wird ein Element `tc:ServiceProvider` erstellt. Prüfen Sie den Wert des Attributs `endpoint`, damit Sie den Alias im richtigen Element ändern!

7. Klicken Sie auf „Fertig stellen“, um die Konfiguration zu beenden.
8. Wenn Ihr Web Service Provider in der inubit Suite 6 erstellt wurde, müssen Sie noch dessen Kommunikation mit dem STS konfigurieren.

   → Siehe *Web Services Provider durch Security Token Service absichern (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.6, S. 424).*

33.3 Dialog „Security Token Service (WS-Trust)“

In diesem Dialog konfigurieren Sie den Security Token Service, registrieren Service-Provider und konfigurieren deren Authentifizierung am STS.

Service-Name

Name, unter dem der STS seinen Dienst anbieten soll.

Standardmäßig wird der Name des Konnektors verwendet. Sie können diesen Wert ändern.

WSDL-Datei

WSDL-Datei mit der Schnittstellenbeschreibung des STS, den Pfaden zum Keystore des STS und zum Truststore der gesicherten Service Provider sowie den dazugehörigen Passwörtern und Aliasen.

Transport-Sicherheit

Zur Auswahl des Verschlüsselungsverfahrens:

- **XML-Encryption**
  Die Daten innerhalb der zu übertragenden Nachrichten werden entsprechend dem Standard „XML-Encryption“ verschlüsselt.

  Siehe http://www.w3.org/TR/xmlenc-core/.
- **SSL**
  Der Datentransport wird entsprechend dem SSL-Protokoll verschlüsselt.

STS-Authentifizierung

Zum Hinzufügen des Keystores:

- **Passwort**
  Passwort des Keystores.
- **Keystore /Button „Datei auswählen“**
  Zum Importieren des Keystores.
  Nach erfolgreichem Import wird die Gültigkeit des Schlüssels angezeigt.

Consumer-Authentifizierung

- **X.509**
  Authentifizierung mit einem X.509-Zertifikat. Die Datei mit den Zertifikaten können Sie über den „Truststore“-Button importieren.
- **Benutzer/Passwort**
  Authentifizierung über eine Benutzer/Passwort-Prüfung, die Anmeldungen gegen die folgenden Benutzerverwaltungen prüft.

Wenn Sie beide Optionen markieren, dann werden beide Benutzerverwaltungen geprüft. Die Prüfung wird als erfolgreich gewertet, wenn die Prüfung in einer der beiden Benutzerverwaltungen erfolgreich ist.

- **Interne Benutzerverwaltung**:
  Die Authentifizierung ist nur für Consumer möglich, die als iS-Benutzer in der inubit BPM-Suite vorhanden sind.
- **Authentifizierung durch Workflow**
  Die Authentifizierung der Consumer erfolgt durch einen Workflow gegen ein Backendsystem, wie z. B. ein LDAP-Verzeichnis. Den Workflow müssen Sie selbst entsprechend erstellen.
  Wenn ein Workflow definiert ist, dann erzeugt das letzte Modul des Workflows eine Ausgangsnachricht mit einer `SAMLAssertion`-Workflowvariable. Diese Variable enthält Authorisierungsdaten. Die Daten werden dem Service Provider übergeben und können für rechteabhängige Implementierungen genutzt werden.

Gesicherte Service-Provider

Zum Registrieren der Web Services:

- **URI des Provider-Endpoints**
  URL des registrierten Web Service.
- **Öffentlicher Schlüssel ist gültig bis:**
  Die Information über die Gültigkeit des öffentlichen Schlüssels ist im X.509-Zertifikat gespeichert, welches beim Registrieren eines Service-Providers hinzugefügt wird.
- **Schlüssel-Informationen**
  Listet alle zusätzlichen Informationen, die neben der Gültigkeit im Zertifikat enthalten sind.

34 Selenium Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Der Selenium Connector verbindet das Test-Framework Selenium mit der Testumgebung der inubit Suite 6.

Er wird verwendet, um automatisierte Testfälle bzw. Test-Suiten webbasierter Benutzeroberflächen und Web-Applikationen, die mit dem Selenium-Framework erstellt wurden, aus dem Workflow heraus zu steuern.

Der Konnektor kann eingesetzt werden, um z. B. Portalfunktionen des inubit Enterprise Portals zu testen.

Benutzereigene Erweiterungen mit der Möglichkeit, eigene Test-Kommandos zu definieren und in der user-extensions.js zu hinterlegen, werden nicht unterstützt.

Konnektortypen

Ein Selenium Connector kann als Medium oder Output Connector verwendet werden:

- **Medium Connector**
  Sendet XML-Testdaten an den Selenium Remote Control-Server und erhält Log-Daten vom Selenium-Server zurück, die an das folgende Modul im Workflow übergeben werden.
- **Output Connector**
  Sendet XML-Testdaten an den Selenium Remote Control-Server und erhält Log-Daten vom Selenium-Server zurück, die an eine Zielapplikation weitergegeben werden.

Voraussetzungen

- Die komplette Selenium-Testumgebung muss installiert sein.
- Ein Selenium Remote Control-Server muss installiert sein.
- Ein Web-Browser muss installiert sein.

Informationen über Selenium finden Sie unter folgenden Links:

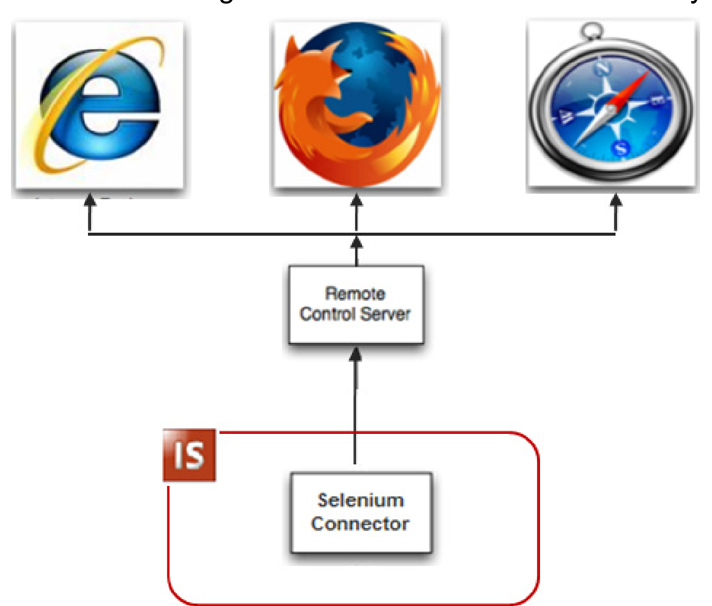
- Selenium-Download: http://seleniumhq.org/download/
- Selenium Remote Control-Server: http://seleniumhq.org/docs/05_selenium_rc.html.
- Selenium Documentation: http://seleniumhq.org/docs/
- Selenium Reference mit allen Test-Befehlen: http://release.seleniumhq.org/selenium-core/0.8.2/reference.html

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

34.1 Funktionsprinzip

Der Selenium Connector verbindet sich mit dem Selenium Remote Control-Server, um einen auf lokalen oder entfernten Rechnern installierten Web-Browser zu steuern. Der Server startet, stoppt den Browser und fungiert für den Browser als HTTP Proxy.



Eingangsnachricht

Der Selenium Connector erwartet eine XML-Eingangsnachricht, die z. B. folgendermaßen aussieht:

```
<?xml version="1.0" encoding="UTF-8" ?>
<script>
  <base-url>http://www.google.de</base-url>
  <suite>
    <command>
      <action>setSpeed</action>
      <target>500</target>
      <value />
```

```
    </command>
    <command>
      <action>open</action>
      <target>/</target>
      <value />
    </command>
    <command>
      <action>type</action>
      <target>q</target>
      <value>inubit</value>
    </command>
    <command>
      <action>clickAndWait</action>
      <target>btnG</target>
      <value />
    </command>
  </suite>
</script>
```

In der Eingangsnachricht muss die `base-url` der zu testenden Applikation angegeben werden: diese URL muss gesetzt werden, um Pfadangaben relativ zu machen, denn so können Tests auch unabhängig vom Ort der Applikation ausgeführt werden.

Für eine korrekte Ausführung müssen die Domains im `base-url`-Element und im `open`-Element identisch sein.

→ Siehe *Fehlerbehandlung (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 34.2, S. 358).*

Templates für Eingabe- und Ausgangsnachrichten finden Sie im Repository unter `Repository > Global > System > Mapping Templates > Selenium Connector`

Verarbeitung

Die Eingangsnachricht übergibt der Selenium Connector dem Selenium Server. Dieser führt die Test-Befehle aus und übergibt dem Connector die Zusammenfassung der Test-Resultate. Diese Log-Daten gibt der Selenium Connector als Ausgangsnachricht aus.

Ausgangsnachricht

```
<result>
  <suite>
    <command display="setSpeed(500)"
duration="31">OK</command>
    <command display="open(/)" duration="813">OK</
command>
```

```
    <command display="type(q)" duration="500">OK</
command>
    <command display="clickAndWait(btnG)"
duration="1062">OK</command>
  </suite>
  <lastlogs>
    <lastlog>
      ![CDATA[13:24:42.413 INFO [10]
org.mortbay.util.Container - Startet … ]]>
    </lastlog>
  […]
  </lastlogs>
</result>
```

34.2 Fehlerbehandlung

Fehler	Ursache	Lösung

You appear to be changing domains from http://XXX to http://ZZZ	Wenn keine Basis-URL angegeben ist, benutzt die inubit Suite 6 die Vorauswahl des RC-Servers.
	Manche Browser werten den Wechsel von der Vorauswahl zu der URL, die im Element `<open>` angegeben ist, als Cross-Site-Scripting und verhindern die weitere Ausführung oder lösen einen Fehler aus.
		Geben Sie die Basis-URL in der Eingangsnachricht im Element `base-url` an, z. B.
		`<base-url>`
		`  http://www.google.com`
		`</base-url>`
		Danach muss das `open`-Element mit der `base-url` identisch sein, da sonst derselbe Fehler wieder auftritt.

Connection refused: connect	Der Remote Control-Server läuft nicht oder die angegebene Server-/Port-Adresse ist falsch.	Prüfen Sie die Verfügbarkeit des Selenium RC-Servers, indem Sie im Dialog auf den „Verbindungstest"-Button klicken. Der Server läuft, wenn Sie als Rückgabe „Verbindungstest war erfolgreich" erhalten und auf dem Rechner, auf dem der RC Server installiert ist, eine Firefox-Instanz geöffnet wird.

Zum Problem der Basis-URL und des Cross-site-Scriptings siehe http://en.wikipedia.org/wiki/Same_origin_policy und http://en.wikipedia.org/wiki/Cross-site_scripting.

34.3 Dialog „Selenium Connector Eigenschaften“

In diesem Dialog geben Sie die Zugangsdaten für den Selenium Remote Control-Server an und konfigurieren Einstellungen für die Verbindung mit dem Server.

Selenium Remote Control-Server

- **Server**
  Adresse des angesprochenen Selenium Remote Control-Servers.
- **Port**
  Der Selenium Remote Control-Server verwendet Port 4444.
- **Browser**
  Zu startender Browser inkl. *.

  Für eine Übersicht über die von Selenium unterstützten Browser siehe http://seleniumhq.org/about/platforms.html#browsers.
- **Timeout (ms)**
  Ausführungszeit in Millisekunden. Nach Ablauf der angegebenen Zeitdauer wird der Test abgebrochen. Standardwert: 30.000 ms.
  Falls hier kein Wert eingegeben ist, fällt der Konnektor automatisch auf den Standardwert zurück.

Ein sehr lange ablaufender Test wird durch die hier gesetzte Timeout-Beschränkung vorzeitig unterbrochen.

- **Screenshot im Fehlerfall erzeugen**
  Wenn markiert, dann wird im Fehlerfall ein Screenshot in Form einer base64-kodierten png-Datei als `error-screenshot`-Element in die XML-Ausgangsnachricht geschrieben.

Sie können z. B. über ein Decoder Modul als nachfolgendes Modul im Workflow die base64-kodierte Screenshot-Datei in ein Binärformat überführen und über einen Output File Connector an eine Zielapplikation übergeben.

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

35 SNMP Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Mit dem SNMP Connector können Netzwerkkomponenten wie Server, Router, Switches oder Drucker von einem zentralen SNMP-Server überwacht und konfiguriert werden. Bei der Überwachung von Netzwerkkomponenten kann die Fehlererkennung und die Fehlerbenachrichtigung konfiguriert werden.

Konnektortypen

Die Funktionen des SNMP Connectors sind von der Konfiguration abhängig:

- **Input Connector**
  Übermittelt statisch OID-Werte-Paare aus einer Tabelle.
- **Medium Connector**
  Übermittelt dynamisch OID-Werte-Paare aus der XML-Eingangsnachricht.

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

35.1 Funktionsprinzip

Viele Netzwerkkomponenten enthalten Programme, über die SNMP Kommunikation möglich ist. Diese sogenannten Agenten, die direkt auf dem überwachten Gerät laufen, können den Zustand des Geräts übermitteln und Daten zur Konfiguration der Netzwerkkomponente empfangen und umsetzen.

Kommunikation

Zur Überwachung von Komponenten in einfachen und komplexen IT-Netzwerken wird ein SNMP-Server eingesetzt. Die Agenten der Netzwerkkomponenten kommunizieren über SNMP mit dem Server. Die Kommunikation über SNMP basiert auf dem verbindungslosen User Datagram Protocol (UDP), das den Datenpaketen von SNMP nur noch die Adressierung hinzufügt und daher kaum Bandbreite benötigt. Standardmäßig kommunizieren die Agenten über Port 161 und der SNMP Manager über Port 162.

Die Struktur der SNMP Datenpakete ist bei der Benutzung des SNMP Connectors transparent. Für die Konfiguration des Connectors müssen nur die Verbindungsdaten, die Art des Datenpakets und die OID bekannt sein.

Datenpakete

Von den sechs verschiedenen Datenpaketen, die im SNMP Standard definiert sind, werden für den SNMP Connector nur vier benötigt. Die Datenpakete sind nach den Methoden benannt, die mit ihnen ausgeführt werden. Der SNMP Connector unterstützt die Operationen GET, GETNEXT, SET und TRAP.

OID

Jeder Agent kennt die Menge der Parameter „seiner" Netzwerkkomponente. Über diese kann entweder eine Statusmeldung erfolgen (Trap) oder die Konfiguration geändert werden. Diese Parameter werden „Managed Objects" genannt. Alle Managed Objects werden in einer tabellenartig angelegten Liste gehalten (nicht in einer Datenbank). Diese Liste heißt Management Information Base (MIB). Jedem Managed Object in der MIB wird eine Object Identifier (OID) zugeordnet.

Der Object Identifier ist eine Zeichenkette, die sich aus Zahlen und Punkten zusammensetzt. Dabei folgt der Aufbau der Zeichenkette einer Struktur, die sich aus dem sogenannten MIB-Baum zusammensetzt.

Siehe http://de.wikipedia.org/wiki/SNMP für eine Abbildung des MIB-Baums.

In diesem Nummerncode gibt es auch einen Teil, der für die Firma steht. Die inubit AG hat die 15899 erhalten, damit lautet der Anfang von OIDs:

- im Klartext: iso.org.dod.internet.private.enterprise.inubit.
- als OID: 1.3.6.1.4.1.15899.

Die OID ist nur bis zum Private Enterprise Code standardisiert. Die weitere Zusammensetzung der OID ist nicht genormt. Es liegt bei den Systemadministratoren, eine firmenspezifische Systematik zu erstellen, die alle Netzwerkkomponenten erfasst und diesen Nummern zuweist. Allen Managed Objects der Geräte muss ebenfalls eine Nummer zugewiesen werden. Die Menge der Managed Objects, also

der Ereignisse, die durch SNMP überwachbar oder konfigurierbar sind, ist geräteabhängig und muss den jeweiligen Herstellerhandbüchern entnommen werden.

OID als Konstante und Variable

Häufig wird den Strukturvorgaben für den Aufbau der OID Zeichenkette aus Zahlen und Punkten nicht gefolgt, da die OIDs nur firmenintern genutzt werden und daher keine eindeutige Abgrenzung gegenüber den IT-Landschaften anderer Firmen bestehen muss.

Soll über SNMP ein Trap oder eine Konfigurationsanweisung gesendet werden, so besteht diese üblicherweise aus einer OID und einem Wert. Variable: Jede OID kann genau die Werte bekommen, die in den Herstellerangaben der Netzwerkkomponente angegeben sind. Konstante: Es gibt auch den Fall, dass nur eine OID ohne Wert übermittelt wird. Dies ist der Fall, wenn ein Managed Object über eine OID nur einen Zustand kommunizieren kann.

Community

SNMP bietet die Möglichkeit, bei der Kommunikation einen sogenannten Community String zu übermitteln. Nur Komponenten, die in derselben Community sind, reagieren auf Traps oder Konfigurationsbefehle. Der Community String wird unverschlüsselt verschickt.

35.2 Beispiel: Medium Connector

Über einen SNMP Medium Connector soll im Workflow über einen Router eine ISDN Verbindung aufgebaut werden. Über diese soll anschließend, am Ende des Workflows, eine Nachricht an einen FTP Connector geschickt werden.

Beim SNMP Medium Connector kann keine Tabelle mit Tupeln aus OID und Wert angegeben werden. Hier werden die entsprechenden Wertepaare in der XML-Eingangsnachricht erwartet.

Der Wert einer OID kann eine der folgenden Angaben sein:

- **keine Angabe**
  Die OID ist selbst die Anweisung zu einer Konfiguration.
- **Ganze Zahl**
  Die Konfiguration ist abhängig vom Zahlenwert.
- **Text**

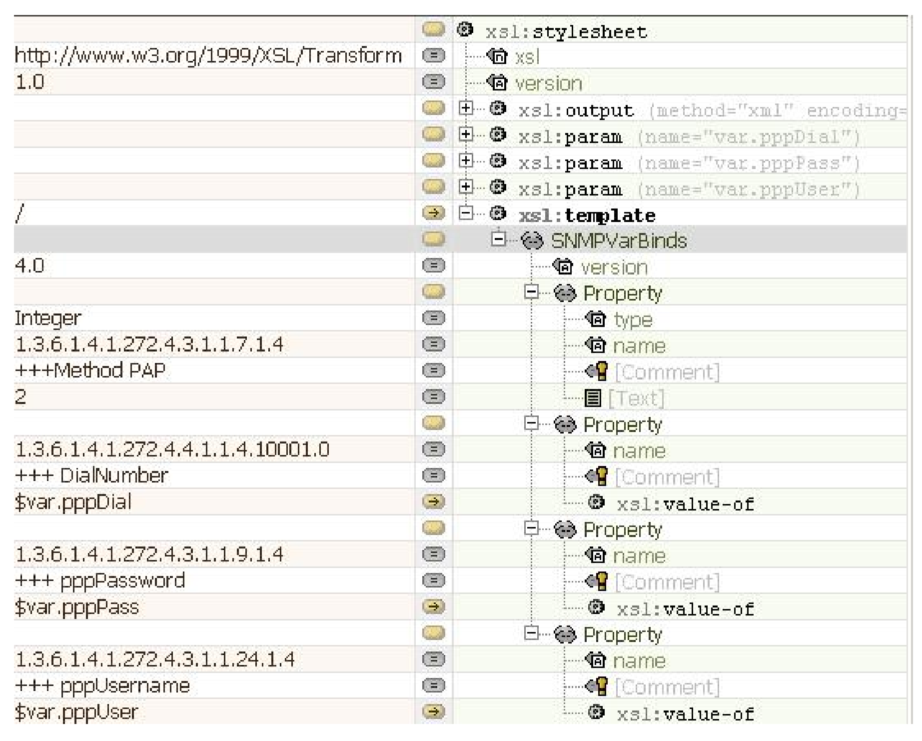
In einer XML-basierten Eingangsnachricht haben diese drei Fälle das folgende Format (OIDs und Werte sind Beispiele):

```
<Property name="1.3.6.1.4.1.272.4.4.1.1.4.
  10001.0"/>
<Property name="1.3.6.1.4.1.272.4.3.1.1.7.1.4"
  type="Integer">
    2
</Property>
<Property name="1.3.6.1.4.1.272.4.4.1.1.4.10001.0">
    UID_name
</Property>
```

Der Router wird dynamisch während der Workflow-Ausführung konfiguriert. Die benötigten Werte werden aus den Eingangsnachrichten gelesen, von einem XSLT Converter in eine XML-Nachricht transformiert und an den SNMP Connector übergeben.

Die folgende Abbildung zeigt das Mapping im XSLT Converter. Dabei bedeuten die Abkürzungen:

- PPP = Point-to-Point Protocol
- PAP = Password Authentification Protocol



Die folgenden Zeilen zeigen die XML-Nachricht, die als Ergebnis das Mappings an den SNMP Medium Connector übergeben wird:

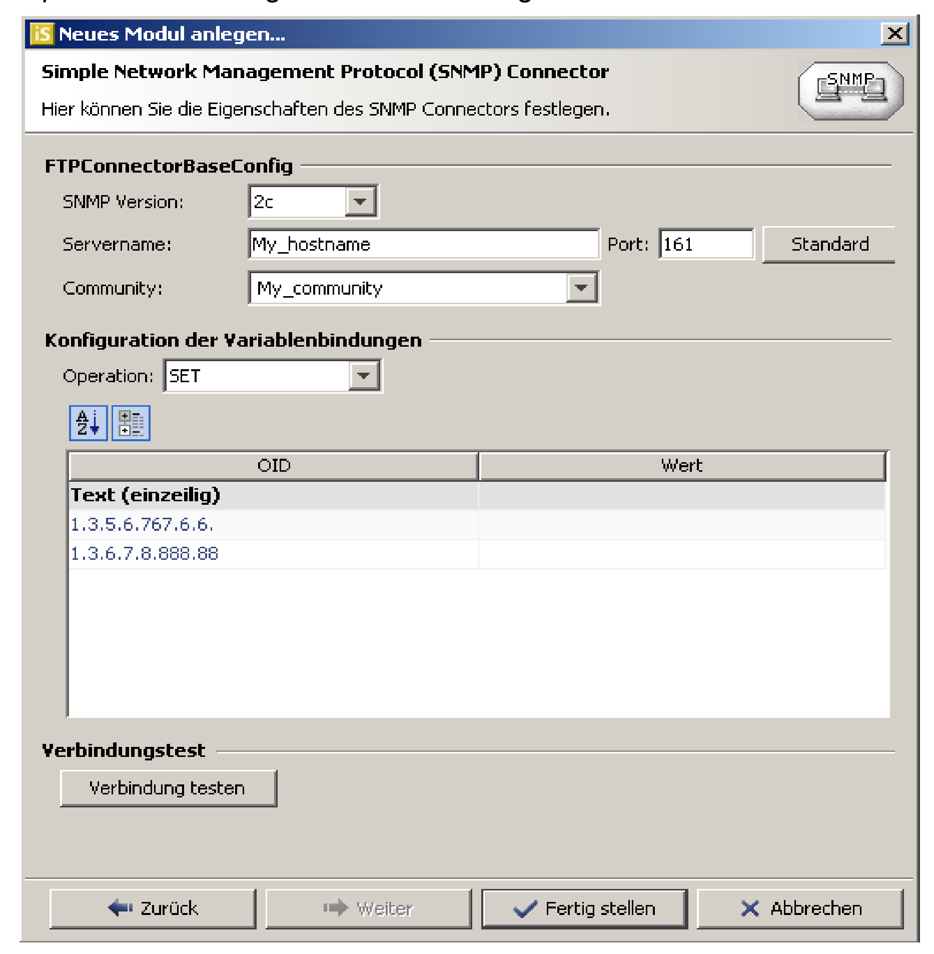
```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<SNMPVarBinds version="4.0">
  <Property name="1.3.6.1.4.1.272.4.3.1.1.7.1.4"
type="Integer">2</Property>
  <Property
name="1.3.6.1.4.1.272.4.4.1.1.4.10001.0"/>
<Property name="1.3.6.1.4.1.272.4.3.1.1.9.1.4"/>
```

```
  <Property name="1.3.6.1.4.1.272.4.3.1.1.24.1.4"/>
</SNMPVarBinds>
```

35.3 Beispiel: Input Connector

Dieses Beispiel bezieht sich auf das Beispiel zum Medium Connector.

Falls die ISDN-Verbindung des Routers immer dieselben Daten benutzt, kann die Verbindung auch am Anfang eines Workflows mit einer statischen OID Tabelle konfiguriert werden. Dazu wird ein SNMP Input Connector folgendermaßen konfiguriert:



35.4 Dialog „SNMP Connector Eigenschaften“

In diesem Dialog haben Sie folgende Optionen:

Grundkonfiguration

- **SNMP Version**
  Mit der Auswahl der Version 1 oder 2c wird die SNMP-Nachricht entsprechend konfiguriert. Fragen Sie Ihren Administrator, welche SNMP Version in Ihrem Firmennetzwerk genutzt wird.
- **Servername**
  Servername als Klarname oder IP-Adresse.
- **Port**
  Standardport ist 161.
  Wenn die SNMP-Kommunikation über einen anderen Port läuft, erfragen Sie die Portnummer bei Ihrem Administrator.
  Der Button „Standard“ stellt die Standardportnummer wieder her.
- **Community**
  Der Community String „public“ wird häufig als Standard verwendet. Sollte ein anderer String verwendet werden, tragen Sie ihn hier ein. Moderne Netzwerkkomponenten können zu mehreren Communities gehören.
  Fragen Sie Ihren Administrator nach dem gültigen Community-String.

Konfiguration der Variablenbindung

Operation

- `GET`: fordert einen Datensatz an. Der Datensatz enthält die Daten zur Konfiguration einer Netzwerkkomponente. Der SNMP Connector, übermittelt als Antwort die Daten, die für dieses Ereignis im Handbuch der Netzwerkkomponente angegeben sind.
- `GETNEXT`: Fordert solange Datensätze an, bis das Ende einer Liste mit Datensätzen erreicht ist. Diese Operation wird benutzt, wenn sich die Konfiguration einer Netzwerkkomponente über mehrere Datensätze erstreckt.
- `SET`: Verschickt einen Datensatz zur Konfiguration einer Netzwerkkomponente. Die Netzwerkkomponenten reagiert mit einem Datensatz, der das Ergebnis der Konfiguration enthält.
- `TRAP`: Automatisch ausgelöste Benachrichtigung über ein Ereignis bei einer überwachten Komponente

Für die Variablenbindung wird eine Tabelle mit Paaren aus OID und Werten aufgebaut. Diese Tabelle ist nur für den Input SNMP Connectoren verfügbar.

Die Tabelle bietet ein Kontextmenü zum Hinzufügen und Löschen von OIDs.

Erfragen Sie die OID und den Wert für die gewünschte Aktion bei Ihrem Administrator erfragen oder entnehmen Sie diese den Handbüchern der jeweiligen Netzwerkkomponenten.

Trap Konfiguration

(Nur bei Auswahl der Operation „TRAP“)

- **Absender IP**
  IP-Adresse oder Hostname des Rechners, der SNMP-Traps sendet.
- **Absender OID**
  OID des Gerätes, welche den Trap generiert. Die OID spezifiziert, um welchen SNMP-Agenten es sich handelt.
  → Für Informationen zum Object Identifier siehe *OID (Workbench/ Process Engine: Systemkonnektor-Guide, Kap. 35, S. 362)*
- **Allgemeine ID**
  Allgemeine Trap-ID. Wählen Sie unter den sieben möglichen Trap-IDs die entsprechende aus.
  Falls es sich um eine firmenspezifische Trap-ID handelt, übertragen Sie diese in das nächste Feld „Spezifische ID“.
- **Spezifische ID**
  Firmenspezifische Trap-ID, falls es sich um eine firmenspezifischen Trap handelt.

Erfragen Sie bei Bedarf die OID und andere Angaben für die Trap-Konfiguration bei Ihrem System-Administrator.

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

36 Solution Center Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Mit dem Solution Center Connector können Sie lesend und schreibend aus Technical Workflows der inubit Suite 6 heraus auf Business Objekte einer Business Solution zugreifen, die in einem Solution Center liegen.

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

Voraussetzungen

- Sie haben das Solution Center erfolgreich installiert und konfiguriert.
- Sie haben die Verbindung der inubit Workbench zum Solution Center konfiguriert.

36.1 Funktionsprinzip

1. Der Solution Center Connector greift lesend oder schreibend auf das Business Objekt in Ihrer Business Solution im Solution Center zu oder legt es neu an.
   - **Ändern, Lesen und Löschen**: Das Business Objekt wird über die Knoten-ID oder eine XPath-Abfrage definiert.

- **Anlegen**: Der Solution Center Connector erhält eine Eingangsnachricht, in der das zu erstellende Objekt definiert ist. Der Vaterknoten, unterhalb dessen das Objekt angelegt werden soll, wird über die Knoten-ID oder eine XPath-Abfrage definiert.

2. Der Solution Center Connector liefert die Daten des Business Objekts mit der übergebenen Knoten-ID bzw. XPath-Abfrage aus der Business Solution zurück. Das Business Objekt kann in nachfolgenden Modulen weiter bearbeitet werden. Durch das Überschreiben von Moduleigenschaften per Variablen-Mapping können Sie u. a. die Knoten-ID aus einer vorhergehenden Objektabfrage setzen.

36.2 Moduleigenschaften

Der Solution Center Connector setzt folgende Moduleigenschaften:

Name	Erläuterung
Request.Method	Modus des Konnektors. Mögliche Werte: ■ GET: Objektdaten abfragen ■ POST: Objekt erzeugen ■ PUT: Objektdaten aktualisieren ■ DELETE: Objekt löschen
NodeID	ID des Knotens (bzw. Vaterknotens beim Erzeugen eines Objekts).
XPath	XPath-Abfrage, um den Knoten (bzw. Vaterknoten) zu bestimmen. Dieser XPath wird auf der Objekt-Struktur des Solution Centers ausgeführt. Üblicherweise geben Sie hier auch eine Kontextknoten-ID an.
ContextNodeID	Nur in Zusammenhang mit dem Parameter „XPath“: ID eines Solution Center-Knotens. Die XPath-Abfrage wird relativ zu diesem Knoten ausgeführt.
createEmptyObject	leeres Objekt anlegen (nur beim Anlegen eines neuen Objekts)
returnOutput	Rückgabe der Knoteninformationen. Über diesen Schalter unterdrücken Sie die Rückgabe der geänderten Objektdaten beim Erzeugen, Aktualisieren oder Löschen eines Objektes.

Name	Erläuterung
OutputDepth	Anzahl der zurückgegebenen Knotenebenen

36.3 Objekt-XML-Daten erzeugen

Beim Erstellen eines Business Object Diagramms (BOD) generiert die inubit Suite 6 automatisch ein XSD-Schema des Diagramms und speichert es im Repository-Ordner „SolutionCenter/BODSchemas/“ des Benutzers, der das BOD erstellt hat.

Diese Schemas können Sie nutzen, um mit einem XSLT Converter eine Eingangsnachricht zum Erzeugen oder Aktualisieren eines Objekts zu erstellen.

36.4 Verwenden von XPath-Anfragen

Verwendung

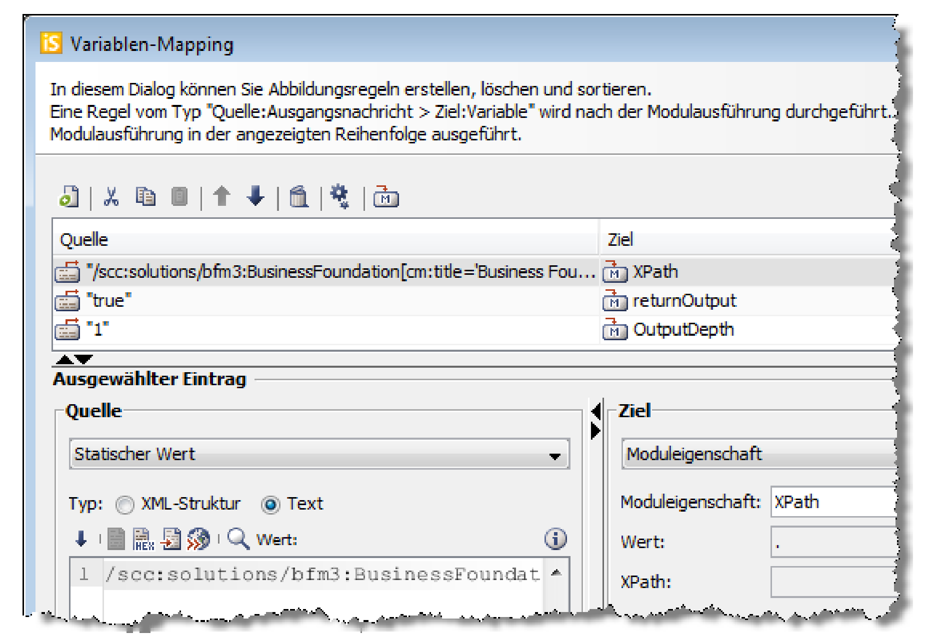
Flexibler Zugriff auf Daten einer Business Solution oder Process Solution im Solution Center.

Aufruf

- XPath-Abfrage über den Solution Center Connector
  Setzen Sie die Parameter auf dem Register SC-Connector Einstellungen.
  - XPath
    Geben Sie den XPath direkt in das Feld XPath ein oder wählen Sie den gewünschten Knoten aus. Der korrekte XPath wird automatisch in das XPath-Feld eingetragen.
  - Kontextknoten verwenden/Knoten-ID
    Optional wählen Sie einen Kontextknoten aus. Die Knoten-ID wird automatisch eingetragen. Die XPath-Abfrage wird ab diesem Knoten ausgeführt.
- XPath-Abfrage über die Moduleigenschaften des Solution Center Connectors
  Konfigurieren Sie den Solution Center Connector.

Setzen Sie die Moduleigenschaften XPath, returnOutput und OutputDepth, um bestimmte Werte über das Variablen-Mapping zu überschreiben.

→ Siehe *Moduleigenschaften (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 36.2, S. 370).*



Ihre XPath-Abfragen nutzen XPath 1.0 mit folgenden Einschränkungen/Empfehlungen:

- Vermeiden Sie in Anfragen möglichst „//“ und Wildcards „/*“.
- Nutzen Sie die Angabe eines Kontextknotens, um die Anfrage zu beschleunigen.

36.4.1 Aggregationen, ein- und zweiseitig gerichtete Assoziationen

Über XPath-Anfragen können Sie neben Aggregationen (Eltern-Kind-Beziehung) auch ein- und zweiseitig gerichtete Assoziationen adressieren.

- Bei einer Aggregation navigieren Sie vom Elternknoten zum Kindknoten.

  Beispiel: `../fpm:Fuhrpark/fpm:Fahrzeug` (alle Fahrzeuge des Fuhrparks)
- Bei einer einseitig gerichteten Assoziation navigieren Sie vom Quellknoten zum Zielknoten.

  Beispiel: `../fpm:Mitarbeiter/fpm:IstFahrerVon` (Mitarbeiter ist Fahrer von Fahrzeug)

- Bei einer zweiseitig gerichteten Assoziation können Sie sowohl vom Quellknoten zum Zielknoten als auch vom Zielknoten zum Quellknoten navigieren.
  Beispiel:
  `../fpm:Mitarbeiter/fpm:IstHatFahrer` (Mitarbeiter ist Fahrer von Fahrzeug)
  `../fpm:Fahrzeug/fpm:IstHatFahrer` (Fahrzeug hat Mitarbeiter als Fahrer)

36.5 Beispiel: Objekt erzeugen

Dieser Abschnitt erläutert die folgenden Themen:

- *Objekt über die Knoten-ID erzeugen, S. 373*
- *Objekt über eine XPath-Anfrage erzeugen, S. 374*

Verwendung Adressieren des Vaterknotens und optional Rückgabe des neuen Knoteninhalts als XML-Struktur

36.5.1 Objekt über die Knoten-ID erzeugen

So gehen Sie vor

1. Erstellen Sie in einem Technical Workflow einen Solution Center Connector und benennen Sie diesen.
2. Wechseln Sie auf die Seite „SC-Connector Einstellungen“.
3. Wählen Sie den Modus „Objekt erzeugen“.
4. Klicken Sie auf den Button „Knoten auswählen“ am Ende des Feldes „Knotens“.
5. Geben Sie den Vaterknoten an, unter dem Sie einen neuen Knoten erstellen wollen.
6. Geben Sie im Feld „leeres Objekt vom Typ“ den Typ des neuen Objekts an oder wählen Sie diesen über den Button „XML-Element auswählen“.

Der Typ des angegebenen Vaterknotens muss es unterstützen, dass Kindknoten des angegebenen Typs unterhalb des Vaterknotens angelegt werden dürfen.

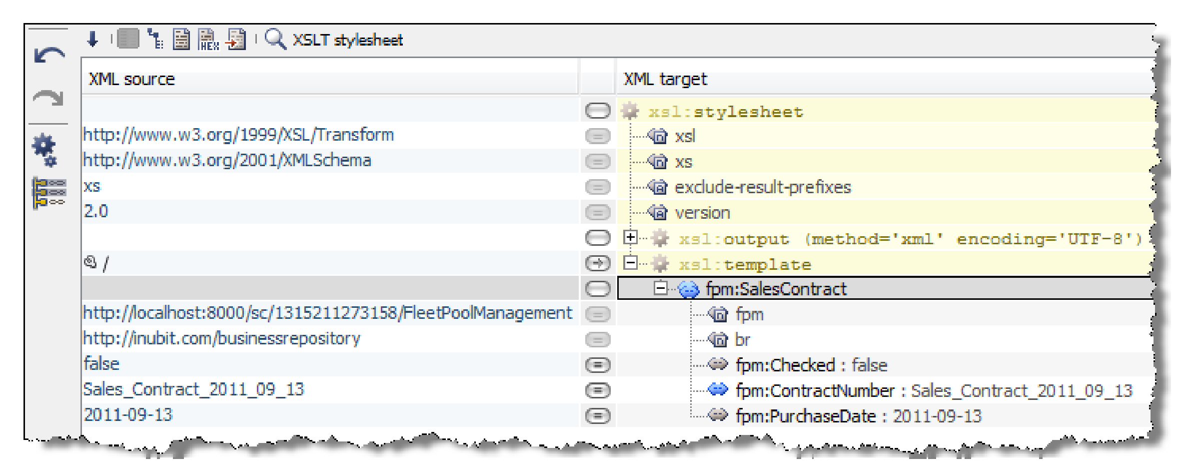
Alternativ dazu können Sie die Attribute des Objekts bereits beim Anlegen über eine Eingangsnachricht definieren.

7. Setzen Sie den Startpoint vor den Solution Center Connector.
8. Starten Sie den Test ohne Datei.

36.5.2 Objekt über eine XPath-Anfrage erzeugen

So gehen Sie vor

1. Erstellen Sie anhand des XSD-Schemas Ihrer Business oder Process Solution eine XML-Struktur, z. B. über einen XSLT Converter.

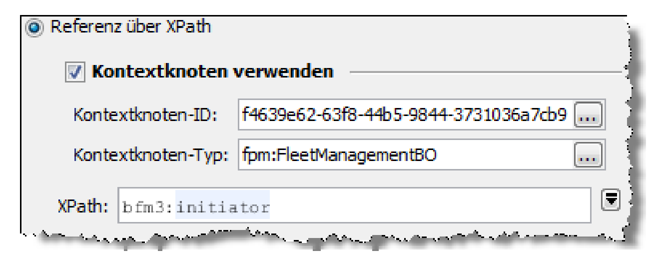


2. Erstellen Sie einen Solution Center Connector.
3. Konfigurieren Sie die allgemeinen Moduleigenschaften und die System Connector Eigenschaften des Solution Center Connectors.
4. Wechseln Sie auf das Register „SC-Connector-Eigenschaften“.
5. Wählen Sie den Modus „Objekt erzeugen“.
6. Geben Sie an, unter welchem Knoten das neue Objekt angelegt werden soll.
7. Wählen Sie die Option „Referenz über XPath“.

XPath mit Kontextknoten

   a. Aktivieren Sie die Option „Kontextknoten verwenden“.
   b. Klicken Sie auf das Icon [...] am Ende der Zeile „Kontextknoten-ID“.
   c. Navigieren Sie im Solution Center Explorer zu dem Vaterknoten, unter dem der neue Knoten angelegt werden soll, und bestätigen Sie die Auswahl mit „OK“.

      Der Kontextknoten-Typ wird abhängig vom gewählten Kontextknoten automatisch gesetzt.

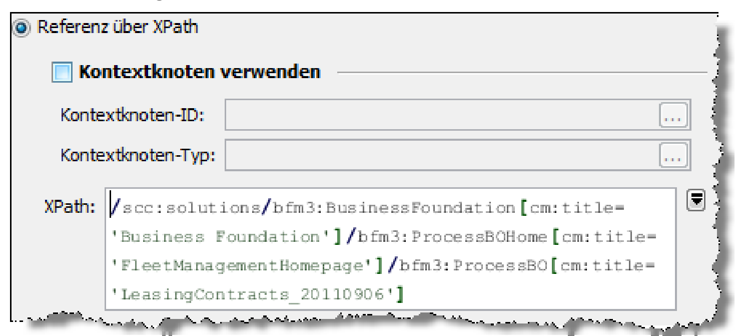


d. Tragen Sie z. B. „`bfm3:Initiator`" in das Feld „XPath" ein.

XPath ohne Kontextknoten

Die Bearbeitung einer Abfrage ohne Kontextknoten kann insbesondere bei großen Datenmengen deutlich länger dauern als eine Abfrage mit Kontextknoten.

a. Deaktivieren Sie die Option „Kontextknoten verwenden".
b. Klicken Sie auf das Icon ⯆ neben dem XPath-Feld und wählen Sie „Knoten auswählen".
c. Navigieren Sie im Solution Center Explorer zu dem Vaterknoten, unter dem der neue Knoten angelegt werden soll, und bestätigen Sie die Auswahl mit „OK".



8. Tragen Sie in das Feld „Anzahl der Knotenebenen" den gewünschten Wert ein.
   → Siehe *Anzahl der Knotenebenen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 36, S. 384)*
9. Setzen Sie den Startpoint vor den XSLT Converter und starten Sie den Test ohne Datei.
10. Prüfen Sie in der Ergebnisdatei oder im Solution Center Explorer, ob das Objekt korrekt angelegt wurde.

36.6 Beispiel: Objektdaten abfragen

Dieser Abschnitt erläutert die folgenden Themen:

- *Objekt über die Knoten-ID erzeugen, S. 373*
- *Objekt über eine XPath-Anfrage erzeugen, S. 374*

Verwendung

Adressieren eines Knotens, eines Attributs oder einer Assoziation und Rückgabe der Knoteninhalte bzw. des Attributs als XML-Struktur bzw. XML-Element mit Datentyp

36.6.1 Objektdaten über die Knoten-ID abfragen

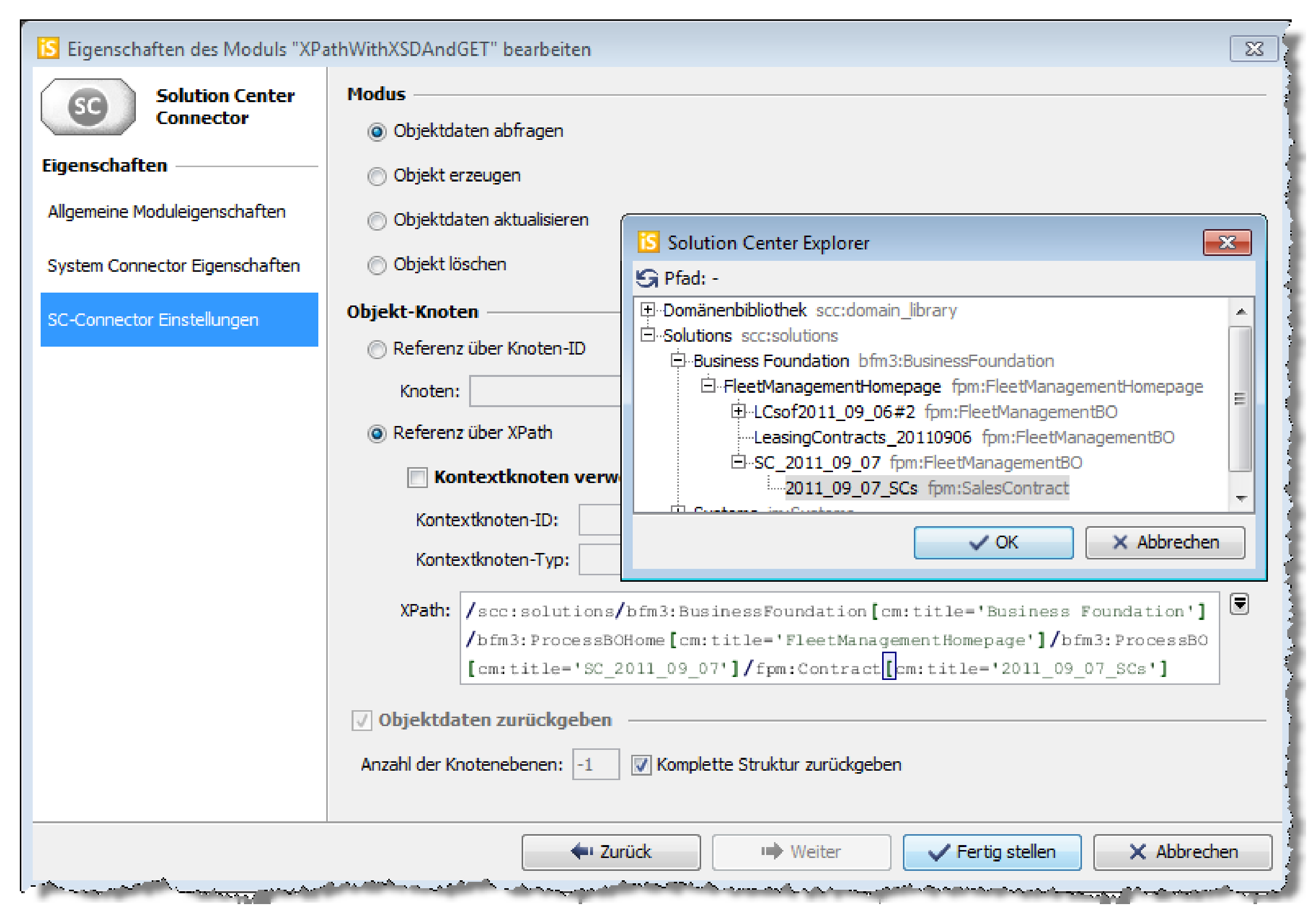
So gehen Sie vor

1. Erstellen Sie in einem Technical Workflow einen Solution Center Connector und benennen Sie diesen.
2. Wechseln Sie auf die Seite „SC-Connector Einstellungen".
3. Wählen Sie den Modus „Objektdaten abfragen".
4. Klicken Sie am Ende des Feldes „Knoten" auf den Button „Knoten auswählen".
5. Navigieren Sie in Ihrer Business Solution zu dem Knoten, den Sie auslesen wollen.
6. Klicken Sie auf „Fertigstellen".
7. Setzen Sie den Startpoint vor den Solution Center Connector.
8. Starten Sie den Test ohne Datei.

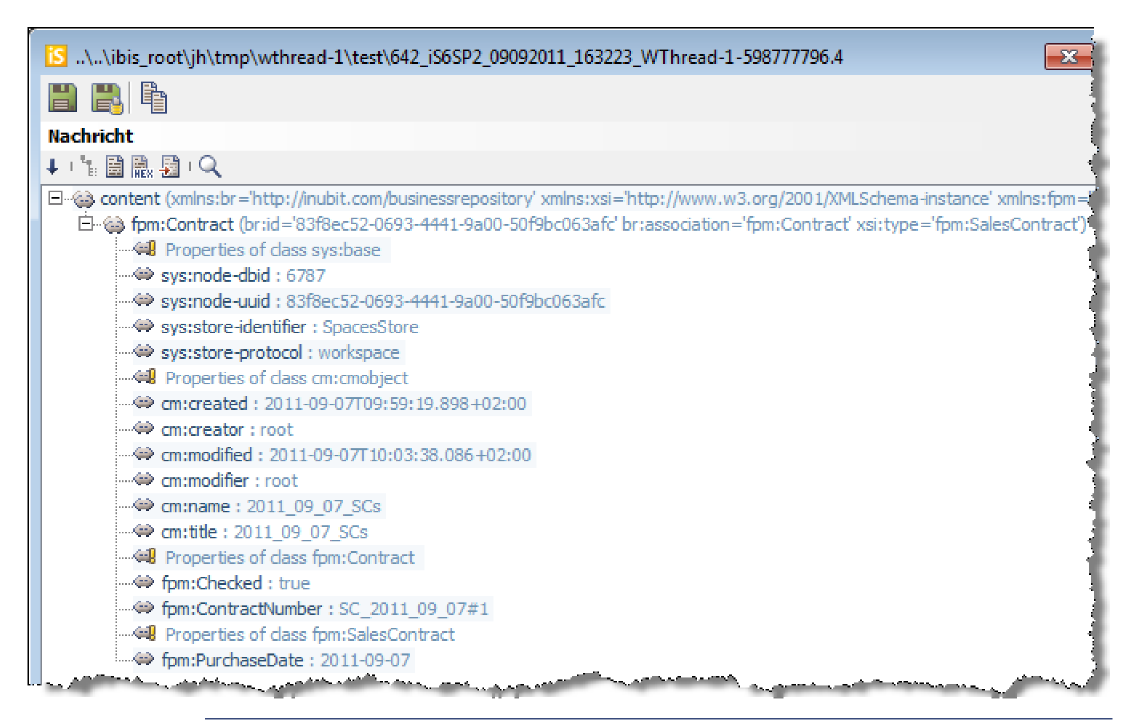
36.6.2 Objektdaten über eine XPath-Anfrage abfragen

So gehen Sie vor

1. Erstellen Sie einen Solution Center Connector.
2. Konfigurieren Sie allgemeinen Moduleigenschaften und die System Connector Eigenschaften des Solution Center Connector.
3. Wechseln Sie auf das Register „SC-Connector-Eigenschaften".
4. Wählen Sie den Modus „Objektdaten abfragen".
5. Klicken Sie auf das Icon ▾ neben dem XPath-Feld.
6. Navigieren Sie im Solution Center Explorer zu dem gewünschten Knoten und bestätigen Sie die Auswahl mit „OK".



7. Klicken Sie auf „Fertigstellen“.
8. Setzen Sie den Startpoint vor den Solution Center Connector.
9. Starten Sie den Test ohne Datei.
10. Öffnen Sie die Ergebnisdatei nach dem Solution Center Connector. Die Ergebnisdatei sollte wie folgt aussehen:



Haben Sie eine Process Solution erstellt und die Klassen im Bereich „Bewegungsdaten" des Process-BODs erstellt, erscheinen die Klassennamen nicht direkt als Tag, sondern im Attribut „xsi:type". Dasselbe trifft für abstrakte Klassen zu.

36.7 Beispiel: Objektdaten aktualisieren

Dieser Abschnitt erläutert die folgenden Themen:

- *Objektdaten über die Knoten-ID aktualisieren, S. 379*
- *Objektdaten über eine XPath-Anfrage aktualisieren, S. 379*

Verwendung

Adressieren eines Knotens und optional Rückgabe des geänderten Knoteninhalts als XML-Struktur

Mit dieser Operation können Sie anhand des XSD-Schemas auch neue Objekte unterhalb des adressierten Objekts und Assoziationen zwischen diesen in einem Schritt anlegen.

Voraussetzungen

- Sie haben die Daten des zu ändernden Knotens z. B. mit einem Solution Center Connector ermittelt.
  → Siehe *Beispiel: Objektdaten abfragen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 36.6, S. 375).*
- Sie haben die Daten als XML-Struktur gespeichert.

36.7.1 Objektdaten über die Knoten-ID aktualisieren

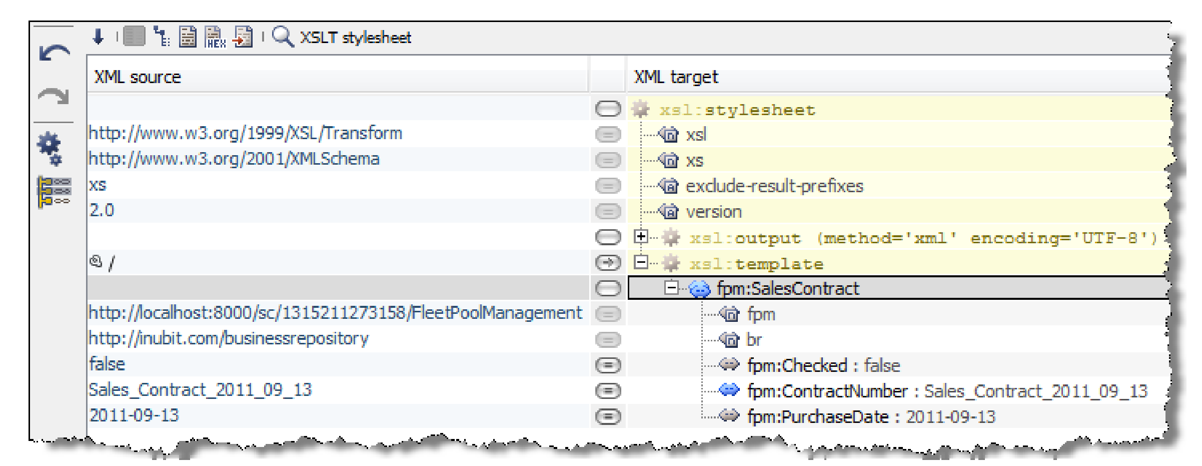
So gehen Sie vor

1. Erstellen Sie in einem Technical Workflow einen Solution Center Connector und benennen Sie diesen.
2. Wechseln Sie auf die Seite „SC-Connector Einstellungen".
3. Wählen Sie den Modus „Objektdaten aktualisieren".
4. Klicken Sie am Ende des Feldes „Knoten" auf den Button „Knoten auswählen".
5. Navigieren Sie in Ihrer Business Solution zu dem Knoten, den Sie aktualisieren wollen.
6. Klicken Sie auf „Fertigstellen".
7. Erstellen Sie eine neue Abbildungsregel für das Variablen-Mapping.
   a. Wählen Sie als Quelle „Statischer Wert".
   b. Öffnen Sie die gespeicherten Daten der Knotenabfrage des zu ändernden Knotens als XML-Struktur.
   c. Bearbeiten Sie die zu ändernden Attribute.
   d. Wählen Sie als Ziel „Eingangsnachricht" und als Kodierung „UTF-8".
   e. Speichern Sie die Änderungen mit „OK".
8. Setzen Sie den Startpoint vor den Solution Center Connector.
9. Starten Sie den Test ohne Datei.

36.7.2 Objektdaten über eine XPath-Anfrage aktualisieren

So gehen Sie vor

1. Erstellen Sie anhand des XSD-Schemas Ihrer Business oder Process Solution eine XML-Struktur mit den Attributen des Knotens, den Sie ändern wollen, z. B. über einen XSLT Converter.



2. Erstellen Sie einen Solution Center Connector.
3. Konfigurieren Sie allgemeinen Moduleigenschaften und die System Connector Eigenschaften des Solution Center Connector.
4. Wechseln Sie auf das Register „SC-Connector-Eigenschaften“.
5. Wählen Sie den Modus „Objektdaten aktualisieren“.
6. Geben Sie den Knoten an, dessen Objektdaten aktualisiert werden sollen.

   → Siehe *Schritt 7* im Abschnitt *Objekt über eine XPath-Anfrage erzeugen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 36, S. 374)*

Wählen Sie anstelle des Vaterknotens den Knoten, den Sie bearbeiten wollen.

7. Aktivieren Sie die Option „Objektdaten zurückgeben“.
8. Tragen Sie in das Feld „Anzahl der Knotenebenen“ den gewünschten Wert ein.

   → Siehe *Anzahl der Knotenebenen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 36, S. 384)*
9. Setzen Sie den Startpoint vor den XSLT Converter und starten Sie den Test ohne Datei.
10. Prüfen Sie in der Ergebnisdatei oder im Solution Center Explorer, ob die Objektdaten korrekt geändert wurden.

36.8 Beispiel: Objekt löschen

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung Adressieren und Löschen eines Knotens und optional Rückgabe der Daten des Vaterknotens des gelöschten Objekts

36.8.1 Objekt über die Knoten-ID löschen

So gehen Sie vor

1. Erstellen Sie in einem Technical Workflow einen Solution Center Connector und benennen Sie diesen.
2. Wechseln Sie auf die Seite „SC-Connector Einstellungen“.
3. Wählen Sie den Modus „Objekt löschen“.
4. Klicken Sie am Ende des Feldes „Knoten“ auf den Button „Knoten auswählen“.
5. Navigieren Sie in Ihrer Business Solution zu dem Knoten, den Sie löschen wollen.
6. Klicken Sie auf „Fertigstellen“.
7. Setzen Sie den Startpoint vor den Solution Center Connector.
8. Starten Sie den Test ohne Datei.

36.8.2 Objekt über eine XPath-Anfrage löschen

So gehen Sie vor

1. Erstellen Sie einen Solution Center Connector.
2. Konfigurieren Sie allgemeinen Moduleigenschaften und die System Connector Eigenschaften des Solution Center Connector.
3. Wechseln Sie auf das Register „SC-Connector-Eigenschaften“.
4. Wählen Sie den Modus „Objekt löschen“.
5. Klicken Sie auf das Icon ▼ neben dem XPath-Feld.
6. Navigieren Sie im Solution Center Explorer zu dem gewünschten Knoten und bestätigen Sie die Auswahl mit „OK“.
7. Tragen Sie in das Feld „Anzahl der Knotenebenen“ den gewünschten Wert ein.
   → Siehe *Anzahl der Knotenebenen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 36, S. 384)*

8. Starten Sie den Test ohne Datei.
9. Prüfen Sie in der Ergebnisdatei, ob die Daten des Vaterknotens korrekt zurückgegeben wurden und im Solution Center Explorer, ob das Objekt korrekt gelöscht wurde.

36.9 Dialog „SC-Connector Einstellungen“

In diesem Dialog konfigurieren Sie den Modus und die Knoten, die abgefragt, erzeugt, geändert oder gelöscht werden sollen.

Modus

- **Objektdaten abfragen**
  Zum Lesen eines Business Objekts einer Business Solution.
- **Objekt erzeugen**
  Zum Erzeugen eines Business Objekts in einer Business Solution.
- **Objektdaten aktualisieren**
  Zum Aktualisieren eines Business Objekts einer Business Solution.
- **Objekt löschen**
  Zum Löschen eines Business Objekts einer Business Solution.

Objekt-Knoten

(Bei Modus = Objektdaten abfragen, Objektdaten aktualisieren, Objekt löschen)

Zum Ermitteln der gewünschten Knoten. Es stehen zwei Verfahren zur Auswahl:

- **Referenz über Knoten-ID**
  Zum Eintragen oder Auswählen der Knoten-ID des Knotens in der Business Solution, den Sie auslesen, aktualisieren oder löschen wollen.
  Im Feld „Knoten“ erscheint anschließend die ermittelte Knoten-ID.
- **Referenz über XPath**

Aus Performance-Gründen sollten Sie einen Kontextknoten verwenden, ab dem die XPath-Suche startet.

  - **Kontextnoten verwenden**
    Aktivieren Sie diese Checkbox, um die Kontextknoten-ID angeben zu können.
    Wählen Sie zuerst die Kontextknoten-ID, damit der Kontextknoten-Typ automatisch gesetzt wird.

- Kontextknoten-ID: ID des Knotens, ab dem die XPath-Abfrage startet.
- Kontextknoten-Typ: Der Typ des Kontextknoten wird automatisch ermittelt, wenn Sie die Kontextknoten-ID ermitteln.

- **XPath**

  XPath-Ausdruck, der auf die XML-Struktur Ihrer Business Solution angewendet wird. Innerhalb des Ausdrucks können Sie Workflow-Variablen verwenden.

  Mit dem XPath-Ausdruck können Sie ein Business Objekt oder ein Attribut eines Business Objekts Ihrer Business Solution referenzieren. Falls Sie einen Kontextknoten angegeben haben, wird der XPath-Ausdruck relativ zu dem angegeben Knoten ausgeführt. Das automatische Generieren des XSD-Schemas für jedes BOD erleichtert das Erstellen neuer Knoten, das Ändern und Löschen vorhandener Knoten und deren Attribute und Assoziationen.

  → Siehe *Verwenden von XPath-Anfragen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 36.4, S. 371).*

Bereich „Zu erzeugendes Objekt“

(Bei Modus = Objekt erzeugen)

Im Bereich „Zu erzeugendes Objekt“ geben Sie an, wie das Objekt angelegt werden soll. Es stehen zwei Verfahren zur Auswahl:

- **aus Eingabenachricht**

  Aktivieren Sie diese Checkbox, wenn Sie das neue Objekt anhand einer an den Solution Center Connector übergebenen XML-Struktur erstellen wollen.
- **leeres Objekt vom Typ**

  Zur Angabe des Typs des neuen, leeren Objekts.

Bereich „Objekt-Vaterknoten“

(Bei Modus = Objekt erzeugen)

Im Bereich „Vaterknoten“ geben Sie den Knoten an, unter dem das neue Objekt angelegt werden soll. Es stehen zwei Verfahren zur Auswahl:

- **Referenz über Knoten-ID**

  Zum Eintragen oder Auswählen der Knoten-ID des Vaterknotens in der Business Solution, unter dem Sie das neue Objekt anlegen wollen.

  Im Feld „Vaterknoten“ erscheint anschließend die ermittelte ID des Vaterknotens.
- **Referenz über XPath**

Aus Performance-Gründen sollten Sie einen Kontextknoten verwenden, ab dem die XPath-Suche startet.

- **Kontextnoten verwenden**
  Aktivieren Sie diese Checkbox, um die Kontextknoten-ID angeben zu können.
  Wählen Sie zuerst die Kontextknoten-ID, damit der Kontextknoten-Typ automatisch gesetzt wird.
  - Kontextknoten-ID: ID des Knotens, ab dem die XPath-Abfrage startet.
  - Kontextknoten-Typ: Der Typ des Kontextknoten wird automatisch ermittelt, wenn Sie die Kontextknoten-ID ermitteln.
- **XPath**
  Mit diesem XPath-Ausdruck können Sie den Vaterknoten eines neuen Business Objekts referenzieren.
  Innerhalb des Ausdrucks können Sie Workflow-Variablen verwenden.
  Falls Sie einen Kontextknoten angegeben haben, wird der XPath-Ausdruck relativ zum angegebenen Knoten ausgeführt.

Bereich „Objekt-Daten zurückgeben“

Geben Sie an, ob Objektdaten zurückgeliefert werden sollen. Im Modus „Objektdaten abfragen“ ist diese Option nicht deaktivierbar.

Es stehen zwei Optionen zur Auswahl:

- **Anzahl der Knotenebenen**
  Geben Sie an, wie viele Knotenebenen unterhalb des erzeugten, aktualisierten oder gelöschten Objekts zurückgegeben werden sollen.
  - 0: Nur die Attribute „`br:id`“, „`br:association`“ und „`xsi:type`“ des bearbeiteten Objekts werden zurückgegeben.
  - 1: Alle Attribute und Properties des bearbeiteten Objekts werden zurückgegeben.
  - 2 .. n: Alle Daten des bearbeiteten Objekts sowie der angegebenen Ebenen unterhalb des bearbeiteten Objekts werden zurückgegeben.
  - -1: wird automatisch gesetzt, wenn die Option „Komplette Struktur zurückgeben“ gewählt wurde.
- **Komplette Struktur zurückgeben**
  liefert die vollständige XML-Struktur aller Objekte zurück.

37 Solution Center Logger

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Erstellen von Ereignisobjekten an bestimmten Stellen einer Process Solution

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

37.1 Funktionsprinzip

- Solution Center Logger einfügen
  - Während des Deployments eines Solution Center Prozessmodell-BPDs wird beim Generieren des Workflows für jedes Start-, Zwischen- und Endereignis je ein Solution Center Logger erstellt.
  - Nach jedem beliebigen Modul können Sie einen Solution Center Logger in den Workflow einfügen.
- Solution Center Logger konfigurieren

  Die Attribute des zu erstellenden Ereignisobjekts konfigurieren Sie über die Moduleigenschaften im Wizard oder über das Variablen-Mapping.
- Automatisches Erstellen von Ereignisobjekten
  - Ein Solution Center Logger erstellt bei jedem Durchlauf ein Ereignisobjekt.
  - Die Ereignisobjekte erscheinen im Solution Center direkt unterhalb des Prozessobjekts der Process Solution.

37.2 Eigene Event-Typen erstellen

So gehen Sie vor

1. Öffnen Sie das zugehörige BOD Ihrer Process Solution zum Bearbeiten.
2. Öffnen Sie das Register „Basis Schemas“ in der Palette „Externe Elemente“.
3. Ziehen Sie eine der Klassen „bfm3:BusinessEvent“ oder „bfm3:Milestone“ in den Rahmen „Bewegungsdaten“.
4. Erstellen Sie weitere Event-Klassen.
5. Verbinden Sie die neuen Klassen mit der jeweiligen Elternklasse („bfm3:BusinessEvent“ oder „bfm3:Milestone“) über eine Generalisierung.
6. Publizieren Sie das BOD.
7. Deployen Sie das BOD in das Solution Center mit einem Klick auf das Icon .

Nach dem Deployment des BODs stehen die neuen Event-Typen im Solution Center Logger zur Verfügung.

→ *Event-Typ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 37, S. 386)*

37.3 Dialog „SC-Logger Einstellungen“

In diesem Dialog konfigurieren Sie den Solution Center Logger.

Ziel-Knoten

- **Knoten-ID**
  Wählen Sie einen Knoten, unter dem das neue Event-Objekt angelegt werden soll. Im Allgemeinen ist dieser Knoten eine Instanz von „bfm3:ProcessBO“.
- **Neuen Status-Inhalt setzen**
  Aktivieren Sie diese Option und tragen Sie die gewünschte Statusmeldung in das nebenstehende Feld ein, um sie dem Status-Attribut des oben angegebenen Knotens zuzuweisen.

Eventdaten

- **Event-Typ**
  - **bfm3:BusinessEvent**
    Erstellt ein Ereignisobjekt der Klasse „bfm3:BusinessEvent“ mit den Attributen „Title“, „Description“, „Actor“ und „Action Date“.

- **bfm3:Milestone**
  Erstellt ein von der Klasse „bfm3:BusinessEvent“ abgeleitetes Ereignisobjekt der Klasse „bfm3:Milestone“ mit den zusätzlichen Attributen „Position“ und „Status“.

→ Siehe *Eventdaten-Mapping (Workbench/Process Engine: Systemkonnektor-Guide, Kap. , S. 387).*

Sie können eigene Event-Typen erstellen.

→ Siehe *Eigene Event-Typen erstellen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 37.2, S. 386).*

Eventdaten-Mapping

- **Title**
  Bezeichnung des Ereignisobjekts, wird aus dem Namen des Ereignisses im BPD übernommen
- **Description**
  Beschreibung des Ereignisobjekts
- **Actor**
  Benutzer, der das Ereignisobjekt erstellt hat
- **Action Date**
  Erstellungsdatum des Ereignisobjekts
- **Position (nur für den Event-Typ „bfm3:Milestone“)**
  Positionsnummer eines Meilensteins
- **Status (nur für den Event-Typ „bfm3:Milestone“)**
  Status des Meilensteins

Für selbstdefinierte Event-Typen können Sie weitere Attribute definieren und über die Moduleigenschaften im Wizard oder das Variablen-Mapping setzen.

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Ein VFS (Virtual File System) Connector sorgt dafür, dass die inubit Process Engine mit lokalen und entfernten Dateisystemen, wie z B. Windows Shared Verzeichnissen oder dem lokalen Repository kommunizieren kann. Der Konnektor unterstützt dafür das CIFS/Samba- und das Repository-Protokoll.

Konnektortypen

Sie haben folgende Konfigurationsmöglichkeiten:

- **Input Connector**
  Holt Dateien aus einer virtuellen Verzeichnisstruktur und übergibt diese zur Weiterverarbeitung an einen Workflow.
- **Output Connector**
  Schreibt das Ergebnis eines Workflows in eine Datei in einem virtuellen Verzeichnis.

→ Siehe auch
  - *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
  - *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
  - *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

38.1 Modulvariablen des VFS Connectors

Bei der Ausführung eines VFS Connectors werden Modulvariablen gesetzt.

Input Connector

Die folgenden Variablen sind verfügbar, wenn eine Datei gelesen wird und wenn als Ausgangsformat „DATA“ gewählt wurde.

- **`ReadFileDate`**
  Letztes Änderungsdatum der Eingangsdatei im Format `yyyy-MM-dd'T'HH:mm:ss`.
- **`ReadFileName`**
  Name der Eingangsdatei mit Extension.

- **`ReadFileSize`**
  Größe der Eingangsdatei, in Bytes.

Output Connector

Die folgenden Variablen sind verfügbar, wenn eine Datei geschrieben wird und wenn als Eingangsformat „DATA“ gewählt wurde.

- **`WriteFileDate`**
  Erzeugungsdatum der Ausgabedatei im Format `yyyy-MM-dd'T'HH:mm:ss`.
  Nur verfügbar beim Output Connector, wenn eine Datei geschrieben wird und wenn als Eingangsformat „DATA“ gewählt wurde.
- **`WriteFileName`**
  Name der Ausgabedatei.
  Nur verfügbar beim Output Connector, wenn eine Datei geschrieben wird und als Eingangsformat „DATA“ gewählt wurde.

→ Für Infos über weitere Variablen und deren Verwendung siehe *Workflow-Variablen und Mappings (Workbench: Benutzer-Guide, Kap. 14, S. 365)*.

38.2 Dialogbeschreibungen

Dieser Abschnitt erläutert die folgenden Themen:

38.2.1 Dialog „VFS Connector Grundkonfiguration“

Im Dialog „Grundkonfiguration“ konfigurieren Sie die Grundeinstellungen des VFS Connectors. Diese Konfiguration müssen Sie sowohl für den Input als auch für Output Connector durchführen.

Protokoll

Protokoll

Mögliche Protokolle sind:

- **cifs**
  (Common Internet File System) Protokoll für den Fernzugriff auf Dateisysteme über ein Netzwerk.
- **repository**
  Für den Zugriff auf das lokale Repository der inubit Suite 6.

Authentifizierung

- **Typ**
  - Wenn Sie für den Zugriff auf den Server keine Authentifizierung benötigen, markieren Sie „Anonym".
  - Wenn Sie als Protokoll „repository" gewählt haben, steht auch der Typ „Mit Benutzer/Gruppe des Workflows" zur Auswahl. Der VFS Connector authentifiziert sich dann mit den Login-Daten des Workflows.
- **Login/Passwort**
  Wenn für Ihren Server eine Authentifizierung notwendig ist, geben Sie hier die erforderlichen Login-Daten ein.

Normalerweise wird die Art der Passwortversendung zwischen Server und Client ausgehandelt. Sie können jedoch bei Bedarf die Unterstützung von Klartextpasswörtern für alle CIFS-Verbindungen manuell deaktivieren. Dazu erstellen Sie ein neues Java System Property namens `jcifs.smb.client.disablePlainTextPasswords` mit dem Wert „true", siehe *Java System Properties (Process Engine: Administrator- und Entwickler-Guide, Kap. 2.17, S. 38)*.
Beachten Sie, dass eine Fehlermeldung geworfen wird, falls nach der Deaktivierung ein Server explizit Klartextpasswörter verlangt!

Server-Konfiguration

- **Windows Domäne**
  Name der Windows-Domäne, falls der Server in einer solchen läuft und Sie sich über eine Windows Domäne anmelden müssen.
- **Servername**
  Domainname des Servers, z. B. xyz.inubit.com
- **Port**
  Die Standardportnummer ist 445.
- **Standard**
  Stellt bei Bedarf die Standardportnummer 445 wieder ein.

38.2.2 Dialog „Input Connector-Konfiguration“

In diesem Dialog konfigurieren Sie, wie der Konnektor Daten einliest.

Verzeichnis

- **Pfad**
  Absoluter Pfad zu dem Verzeichnis, aus dem Dateien gelesen werden sollen.
  Bei Zugriff auf das Repository geben Sie den Pfad ein, der im Repository im Register „Detaildaten“ angegeben ist.
  → Siehe *Detaildaten (Workbench: Benutzer-Guide, Kap. 19.11.2, S. 502).*
  Sie können folgende Wildcards verwenden, um z. B. Dateien aus Verzeichnissen mit sich ändernden Namen und in beliebiger Verzeichnistiefe abholen zu lassen:
  - * (ein Asterisk): Zum Filtern des Verzeichnisnamens oder Teilen davon. Sie können in einem Namen auch mehrere Wildcards verwenden. Beispiele:
    - `/*/bar` durchsucht /foo/bar
    - `/*oo/bar` durchsucht `/foo/bar` oder auch `/goo/bar`
    - `/*o*/bar` durchsucht `/foo/bar` oder auch `/goa/bar`
  - ** (zwei Asterisks): Zum Filtern des Verzeichnisnamens oder Teilen davon in beliebiger Schachtelungstiefe:
    `/**/bar` durchsucht `/foo/bar` und `/xyz/gfd/bar`

Datei

- **Name**
  Dateiname.
- **Mit Wildcard**
  Wenn markiert, dann können Sie beim Eingeben des Dateinamens Wildcards (Asterisk *) verwenden.
- **Mit regulärem Ausdruck**
  Wenn markiert, dann können Sie mit Hilfe eines regulären Ausdrucks ein Muster angeben, dem die Dateinamen der einzulesenden Dateien entsprechen müssen.

Für Informationen über reguläre Ausdrücke siehe z. B. http://www.perl.com/doc/manual/html/pod/perlre.html oder http://www.regular-expressions.info/tutorial.html.

- **Auswahl umkehren**
  Wenn markiert, dann wird das Muster, das für den Dateinamen durch Wildcard oder regulären Ausdruck angegeben wurde, umgekehrt. Damit werden alle Dateien gelesen, deren Name dem Muster nicht entspricht.
- **Dateien nach dem Lesen löschen**

Wenn markiert, dann werden die gelesenen Dateien nach dem Lesen gelöscht.

- **Max. Anzahl v. Ausführungen pro zeitgesteuertem Aufruf**

  Mit der Angabe legen Sie fest, wie oft pro auslösendem Event der Workflow maximal gestartet werden soll. Auslösendes Event ist das Erreichen eines Zeitpunktes. Dann wird der Workflow so oft gestartet, bis entweder

  - keine weiteren Daten zu holen/zu senden sind oder
  - die Anzahl erreicht wurde, welche bei „maximale Ausführungen pro Aufruf" konfiguriert wurde.

  Mit der Begrenzung der Ausführung steuern Sie die Systemauslastung der inubit Process Engine.

  Die Angabe wird im Test-Modus ignoriert, dann wird der Workflow genau einmal gestartet.

Die Option ist nur bei aktiviertem Scheduler aktiv!

Lesereihenfolge

Wenn mehr als eine Datei übertragen wird, legen Sie fest, ob die Dateien sortiert eingelesen werden sollen und wenn ja, in welcher Reihenfolge.

Beschränkungen

- **Max. Anzahl Dateien**

  Maximale Anzahl von Dateien, die eingelesen werden. Der Standardwert ist 1.

- **Max. Gesamtgröße**

  Kumulative Maximalgröße, bis zu der Dateien pro Ausführung eingelesen werden.

  Wenn aktiviert, dann liest der VFS Connector solange Dateien ein, bis kumulativ der gewählte Maximalwert erreicht ist.

  Wenn als Ausgangsformat „DATA“ gewählt wurde, wird grundsätzlich nur eine Datei eingelesen und dies solange, bis die angegebene Maximalgröße für diese eine Datei erreicht ist.

  Standardwert ist 10 MB.

Falls bei den Beschränkungen Werte im Bereich „kleiner oder gleich Null“ gewählt werden, dann gibt es keine Beschränkung!

Konfiguration des Ausgangsformates

- **DATA**

  Die Ausgangsnachricht enthält Daten, die unverändert weitergegeben werden sollen.

- **XML**

Legt fest, dass die Ausgangsnachricht im IBISDirectory XML-Format weitergegeben wird.

Das XML Schema finden Sie im iS-Repository im Verzeichnis „Global > System > Mapping Templates > File Connector“.

- **ZIP**
  Die Ausgangsnachricht wird in einer Zip-Datei komprimiert.

Hinweis zur XML/Zip-Ausgabe:
Wenn Sie Wildcards im Verzeichnisnamen konfiguriert haben, dann werden die gelesenen Dateien im XML bzw. Zip als Baumstruktur ausgegeben. Als Basisverzeichnis dient dabei das letzte feste Verzeichnis. Beispiele:
Der Verzeichnisname ist `/foo/bar/*/blub`. Es werden die Unterverzeichnisse ab `/foo/bar` abgebildet.
Der Verzeichnisname ist `/foo/bar/xyz/blub`. Es wird das Verzeichnis `/xyz/blub` abgebildet.

XML-Konfiguration

(Bei Ausgabeformat „XML“)

Inklusive Dateiinhalt: Wenn markiert, dann wird die gesamte Datei gelesen und ihr Inhalt base64-kodiert der XML-Ausgangsdatei mitgegeben. Wenn nicht markiert, dann werden nur die Metadaten der Datei gelesenen, z. B. Name, Dateigröße und Datum.

ZIP-Konfiguration

(Bei Ausgabeformat „ZIP“)

Kodierung: Die Auswahl legt die Kodierung der Dateinamen innerhalb des Zip-Archivs fest.

38.2.3 Dialog „Output Connector-Konfiguration“

In diesem Dialog legen Sie fest, wie der Konnektor Daten ausgibt.

Eingangsformat

- **DATA**
  Die Eingangsnachricht besteht aus Daten, die unverändert weitergegeben werden sollen. Die ist z. B. der Fall, wenn die Eingangsdaten als XML vorliegen.
- **XML**
  Die Eingangsnachricht liegt im IBISDirectory XML-Format vor.
  → Siehe *Konfiguration des Ausgangsformates (Workbench/ Process Engine: Systemkonnektor-Guide, Kap. 38, S. 393)*

- **ZIP**
  Die Eingabedaten sind in einer Zip-Datei komprimiert.

Kodierung

(Bei Eingangsformat „ZIP“)

Kodierung: Die Auswahl legt die Kodierung der Dateinamen fest.

Verzeichnis

- **Pfad**
  Absoluter Pfad zu dem Verzeichnis, das der VFS Connector beim Schreiben von Dateien verwenden soll, nach dem Muster `/Root/XY_TestGruppe/XY_Testdaten/`.
- **Erzeuge fehlende Verzeichnisse**
  Erzeugt alle nicht-existierenden Verzeichnisse aus der Pfadangabe.

Datei

(nur bei Eingangsformat „DATA“)

- **Name**
  Dateiname. Optional mit Wildcard bzw. regulärem Ausdruck.
- **Bestehende Datei überschreiben**
  Markiert: Wenn bereits eine Datei mit demselben Namen existiert, dann wird diese überschrieben.
- **Daten an die Datei anfügen**
  Markiert: Die zu lesende Datei wird bei jeder Ausführung des Workflows an eine evtl. bereits existierende Datei angehängt. Dies ist z. B. sinnvoll, um Log-Daten kumulativ in einer Datei zu protokollieren.

Diese Option kann nicht verwendet werden, wenn im Dateinamen Wildcards angegeben sind!

- **Fehler erzeugen, wenn Datei existiert**
  Wenn markiert, dann wird im Queue Manager ein Fehlereintrag für den Workflow mit diesem VFS Connector angezeigt, falls bereits eine Datei mit dem gleichen Namen existiert. Ist ein Fehlerausgang konfiguriert, wird dieser durchlaufen. Sonst wird die Ausführung des Moduls abgebrochen.

39 Web Application Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Der Web Application Input Connector macht die Funktionalität eines Technical Workflows in einem Portalserver als Portlet verfügbar.

Beim Konfigurieren des Systemkonnektors legen Sie folgendes fest:

- Titel und Timeout des Portlets
- Trace Level
- JavaScript-Funktionen und CSS-Klassen, die in allen Formularen verfügbar sind, die im Technical Workflow erzeugt werden
- Rechte des Portlets

Hohe Benutzerzahlen und High Availability sind möglich, indem Sie das Loadbalancing über mehrere identische inubit Process Engines aktivieren und konfigurieren.

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

39.1 Modulvariablen des Web Application Connectors

Ein Web Application Connector setzt folgende Modulvariablen:

Name	Erläuterung
`ISPortalUser`	Loginname des aktuell angemeldeten Portalbenutzers.
`ISWebAppTimeout`	Timeout in Sekunden; wird festgelegt im *Dialog „Web-Applikation“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 39.3.1, S. 400)* bei der Option *Timeout (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 39, S. 401)*
`ISWebAppName`	Name des Web Application Connectors

→ Siehe auch *Modulvariablen definieren (Workbench: Benutzer-Guide, Kap. 14, S. 369)*

Zusätzlich gibt ein Web Application Connector Informationen über Portlet-Instanzen aus, siehe *Informationen aus Portlet-Instanzen abfragen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 39.2, S. 398)*.

39.2 Informationen aus Portlet-Instanzen abfragen

Sie können von jeder Portlet-Instanz folgende Informationen abfragen:

- **ISSession-ID**
  Eindeutige ID der aktuellen Session.
- **ISPortletNamespace**
  Eindeutige ID der verwendeten Portletklasse.
- **ISUserName**
  Name des Benutzers, welcher in der inubit Workbench im Register „Configuration > Allgemein“ im Konfigurationsbereich „Portal > Servereinstellungen“ bei der Option „Portal Login“ angegeben ist.
- **ISPortletActionUrl**
  Eindeutige ID der Portletinstanz.

Die Portletinstanz ist z. B. nötig, um ein mandantenfähiges Portlet zu erstellen, das in Abhängigkeit von der Gemeinschaft, für die es angezeigt wird, unterschiedliche Systeme anspricht. Dazu muss der Technical Workflow, der das Portlet realisiert, wissen, in welchem Kontext die Portlet-Instanz aufgerufen wurde.

- **ISWebLanguage**
  Sprache, die in der Portletinstanz genutzt wird.
- **ISPortalUser**
  Loginname des aktuell angemeldeten Portalbenutzers.
- **ISWebContextPath**
  Aktueller virtueller Pfad des Portlets.
- **ISParameters**
  Liste der URL-Parameter, die an das Portlet übergeben wurden.

Die Informationen werden in der Eingangsnachricht des Web Application Connectors gesetzt und als XML-Struktur ausgegeben.

Voraussetzungen

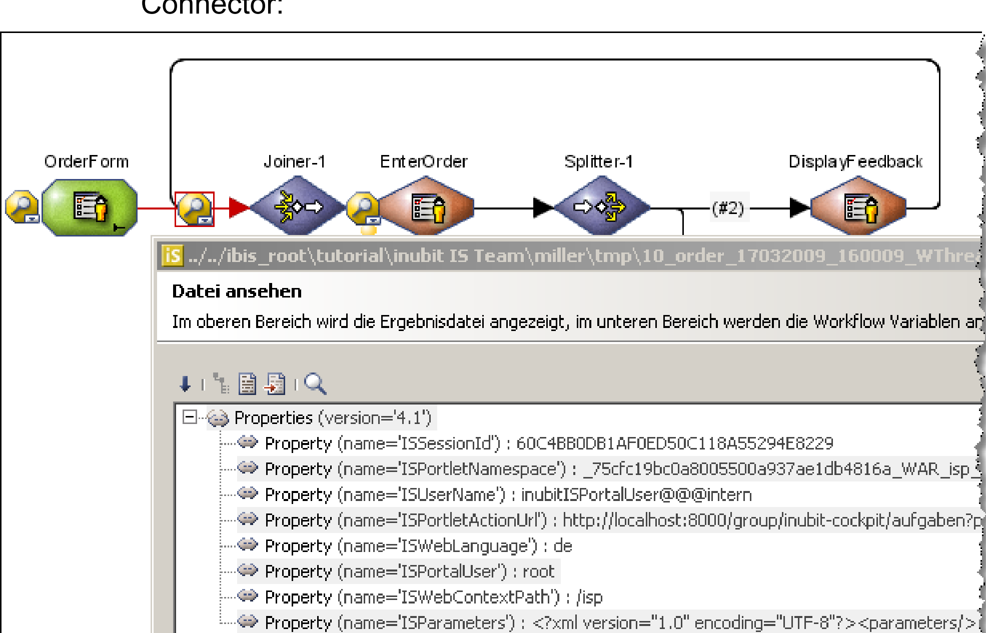
Der Technical Workflow, der das Portlet realisiert, ist bereits deployed.

So gehen Sie vor

1. Zeigen Sie den Technical Workflow, der das Portlet realisiert, in der inubit Workbench an.
2. Schalten Sie den Watch-Modus ein.
   → Siehe *Watch-Modus verwenden (Workbench: Benutzer-Guide, Kap. 16.2, S. 426).*
3. Fügen Sie das Portlet der gewünschten Seite und Gemeinschaft hinzu.

In der inubit Workbench werden Watchpoints angezeigt.

4. Öffnen Sie den Watchpoint vor oder nach dem Web Application Connector:



Die Informationen über die Portletinstanz werden angezeigt.

39.3 Dialogbeschreibungen

Dieser Abschnitt erläutert die folgenden Themen:

39.3.1 Dialog „Web-Applikation“

In diesem Dialog haben Sie folgende Optionen:

Allgemeine Einstellungen

- **Kategorie**
  Legt fest, in welcher Anwendungskategorie das Portlet im inubit Enterprise Portal angeboten wird.

Wenn Sie ein Portlet zu einer Seite hinzufügen, dann wählen Sie es aus einer Liste von Kategorien aus.

- **Standardtitel des Portlets**
  Zum Festlegen des Portlet-Titels in den Standard-Sprachen Deutsch und Englisch.
  Weitere Sprachen fügen Sie über das Kontextmenü der Tabelle hinzu.
- **Timeout**
  Zum Definieren der Zeitbeschränkung, nach deren Ablauf inaktive Benutzer sich erneut am Portlet anmelden müssen.

Erweiterte Einstellungen

- **JSR-168 CSS Klassen verwenden**
  Wenn die Option markiert ist, dann erhalten die CSS-Klassen der Portlet-Komponenten die Namen, die in der JSR 168 definiert sind. Die Portlet-Komponenten passen sich dann automatisch an das Theme an, das aktuell im Enterprise Portal ausgewählt ist, weil für die Darstellung der Themes die JSR 168-konformen Klassennamen genutzt werden. Portlet-Komponenten sind z. B. Buttons, Gruppen oder Eingabefelder.
  Wenn die Option nicht markiert ist, dann werden inubit-spezifische CSS-Klassennamen verwendet.

Siehe http://jcp.org/aboutJava/communityprocess/final/jsr168/index.html, Abschnitt „CSS Style Definitions“.

- **CSS einbinden**
  Wenn die Option markiert ist, dann werden die CSS-Klassen der inubit Suite 6 dynamisch eingebunden (`Layout.css` und `CustomLayout.css` im Verzeichnis `<iS-installdir>/server/ibis_root/conf/form/`).
  Wenn die Option nicht markiert ist, dann müssen Sie CSS-Klassen manuell in den Header der angezeigten Portalseite einbinden.

inubit AG empfiehlt, diese Option zu markieren, wenn Sie den Liferay Portalserver nutzen!

- **JavaScript einbinden**
  Wenn die Option markiert ist, dann werden die mitgelieferten JavaScript-Dateien der inubit Suite 6 dynamisch eingebunden (`Script.js`, `GcScript.js` und `AjaxScript.js` im Verzeichnis `<iS-installdir>/server/ibis_root/conf/form/`).
  Wenn die Option nicht markiert ist, dann müssen Sie JavaScript-Dateien manuell in den Header der angezeigten Portalseite einbinden.

inubit AG empfiehlt, die Option zu markieren, wenn Sie den Liferay-Portalserver nutzen!

- **Formulare in Session nicht cachen**
  Reduziert den Speicherverbrauch in der Portlet-Session.
- **Trace Level**
  Zum Festlegen der Protokollierungstiefe.
  Portlets schreiben ihre Informationen in Protokolldateien im Verzeichnis `<iS-install-dir>ibis_root/log`. Mit Hilfe der Protokolldateien können Sie Portletfehler und außergewöhnliche Bedingungen untersuchen, Portlets testen und Fehler beheben.
- **Interportletkommunikations-Id**
  Eindeutiger Bezeichner für das Portlet, das Sie aktuell konfigurieren. Die ID ist nötig, damit das Portlet bei der Kommunikation zwischen Portlets identifiziert werden kann.
  Diese ID wird automatisch beim Anlegen des Web Application Connectors vergeben, kann aber manuell überschrieben werden.
- **Interportletkommunikation Ziel**
  Eindeutiger Bezeichner des Portlets, dem die Daten des aktuellen Portlets (festgelegt im Feld „Interportletkom. ID) zur Verfügung stehen sollen.
  Daten aus einem Portlet können für alle Portlets oder für ein einzelnes Portlet auf derselben Portalseite verfügbar gemacht werden. Um die Daten aller Portlets auf derselben Portalseite zugänglich zu machen, geben Sie hier keine ID an.
- **Parameter-Trennzeichen**
  Standardmäßig werden Parameter/Werte-Paare in einer URL durch die Zeichen **?** und **&** getrennt, z. B. `inubit.com/search?Parameter1=Wert1`**&**`Parameter2=Wert2`
  Falls Sie einen anderen Portalserver als Liferay verwenden und dieser ein anderes Trennzeichen fordert, geben Sie das Zeichen ein.
- **Lastverteilung**
  Verteilt die Last der Workflow-Ausführung auf mehrere inubit Process Engines, wenn hier die URLs von deren SOAP Servlets angegeben sind.
  Die URLs der SOAP Servlets haben folgendes Muster `http://<rechnername>:8000/ibis/servlet/IBISSoapServlet` und entsprechen den URLs, die beim Login an der inubit Workbench angezeigt werden.
  Auf allen inubit Process Engines muss der Technical Workflow vorhanden sein, den der aktuelle Web Application Connector aufruft.

Die Ausführung der Workflows wird im Round-Robin-Verfahren auf alle angegebenen Rechner verteilt. Dabei wird für eine neue Session der jeweils nächste Rechner gewählt.

→ Siehe

- *Server Trace Log aktivieren und konfigurieren (Process Engine: Administrator- und Entwickler-Guide, Kap. 2.1.3, S. 23)* für mehr Infos über Logging-Level.
- *Daten zwischen Web-Applikationen austauschen: Inter-Portlet-Kommunikation (Workbench: Benutzer-Guide, Kap. 20.17, S. 544)*

39.3.2 Dialog „Interne Ressourcen“

In diesem Dialog haben Sie folgende Optionen:

- **Ressource**

  Aus der Liste können Sie eine der folgenden Optionen auswählen:

  - **LoginPage/ErrorPage**

    Zeigt die mitgelieferte Standard-Fehler- oder Login-Seite für Web-Applikationen an. Sie können die Seiten bearbeiten und bei Bedarf die Standard-Versionen wiederherstellen.

    Der Einsatz der LoginPage ist nur sinnvoll, wenn Sie kein Enterprise Portal einsetzen. Bei Einsatz des Enterprise Portals wird das Login durch das Portal realisiert.

  - **CustomScript**

    Zeigt einen einfachen Editor an, in dem Sie JavaScript-Code eingeben können. Der JavaScript-Code kann in allen Task Generator-Formularen genutzt werden, die den Web Application Connector im *Dialog „Berechtigungen“ (Workbench/Process Engine: Modul-Guide, Kap. 4.10.2, S. 127)* referenzieren.

    → Die Dokumentation des inubit JavaScript-Frameworks finden Sie im Verzeichnis `<iS-installdir>/documentation/jsdoc/index.html`, wenn Sie die inubit-Dokumentation installiert haben.

    Oder rufen Sie die Dokumentation direkt auf.

  - **CustomLayout**

    Zeigt einen Editor an, in dem Sie CSS-Klassen definieren können. Die CSS-Klassen können in allen Task Generator-Formularen genutzt werden, die den Web Application Connector im *Dialog „Berechtigungen“ (Workbench/Process Engine: Modul-Guide, Kap. 4.10.2, S. 127)* referenzieren.

Informationen über die Syntax von Formatdefinitionen für Klassen finden Sie unter http://de.selfhtml.org/css/formate/zentrale.htm#klassen.

→ Siehe *Formulare mit CSS gestalten (Workbench/Process Engine: Modul-Guide, Kap. 4.2.3, S. 65).*

39.3.3 Dialog „Rechteverwaltung“

In diesem Dialog definieren Sie Rechte.

Rechte dieser Web-Applikation

Um ein Recht zu erzeugen, geben Sie eine Zeichenkette in das Eingabefeld ein und klicken auf den Plus-Button. Das Recht wird im oberen Bereich angezeigt. Zum Löschen markieren Sie das Recht und klicken auf den Minus-Button. Das Recht wird aus dem Anzeigebereich entfernt.

Die Rechte werden in Formularen einzelnen Bedienelementen wie z. B. Buttons, DropDown-Listen oder Eingabefelder zugewiesen, beim Deployen des Web Application Connector an das inubit Enterprise Portal übermittelt und dort Rollen zugeordnet. Damit können Sie feingranular steuern, welche Rollen zum Anzeigen und Bearbeiten ausgewählter Formularelemente berechtigt sind.

→ Siehe *Formularelemente abhängig von Rechten machen (nur bei Web-Applikationen) (Workbench/Process Engine: Modul-Guide, Kap. 4.2.7, S. 71).*

40 Web Services Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Der Web Services Connector ist ein zentraler Bestandteil einer SOA. Die Einsatzmöglichkeiten eines Web Services Connectors sind abhängig von dessen Konfiguration:

- **Input Listener Connector**

  Bietet die Funktionalität eines Technical Workflows nach außen als Web Service an.

  Als Input Listener ist der Web Services Connector geeignet zur Integration von Legacy Systemen, die typischerweise nicht über Web Service-Schnittstellen verfügen: dazu wird die Schnittstelle des Legacy Systems mit Hilfe eines Technical Workflows in einen Web Service transformiert und durch den Input Listener als Service verfügbar gemacht.

  Der Input Listener erhält einen Request, startet den Workflow, wartet auf das Ergebnis des Workflows und sendet dann das Ergebnis und die HTTP-Status-Meldung „200“ zurück.

  → Siehe *Web Service anbieten (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.3, S. 409).*

- **Medium Listener Connector**

  Wartet auf die Antwort eines asynchron aufgerufenen Web Services.

  Ein Medium Listener Connector wird daher im Workflow stets hinter einem Medium Connector, der einen asynchronen Web Service aufruft, genutzt.

  → Siehe *Asynchronen Web Service aufrufen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.5.2, S. 420).*

- **Medium/Output Connector**
  Der Connector sendet einen Request an einen externen Web Service oder an einen Web Service Input Listener und wartet auf die Response.
  Sobald der Connector die Response erhalten hat, übergibt er diese an das nächste Modul in seinem Workflow und füllt die Variablen `ISHttpResponseCode` (z. B. mit Wert „200“, wenn OK) und `ISHttpResponseText`.

Die Liste der „OK codes“ können Sie vor Ausführung des Connectors selbst definieren, indem Sie im Register „Eigenschaften“ des Moduls das Element `<Property name="ISHttpStatusCodesOk">` hinzufügen und als Inhalt alle HTTP-Codes eingeben, die **nicht** zu einer Exception führen sollen. Geben Sie die HTTP-Codes als Komma-separierte Liste ein.

Metro

Die Web Services Technologie der inubit Suite 6 basiert auf dem Web Service Stack Metro. Metro ist ein frei verfügbares Produkt des Sun Developer Networks.

Siehe auch http://java.sun.com/webservices/index.jsp

40.1 Funktionsprinzip eines Web Services

Web Services sind Applikationen, welche offene, XML-basierte Standards und Transportprotokolle nutzen, um Daten mit Clients auszutauschen.

Web Services anbieten/abrufen

Serviceanbieter stellen eine Beschreibungen ihrer Web Services als WSDL-Datei zur Verfügung.

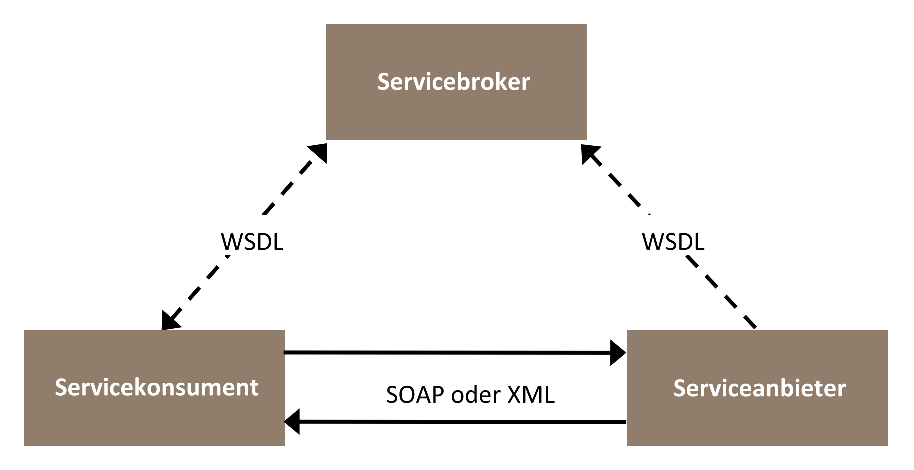
Die WSDL-Datei beschreibt die Operationen, Daten und Austauschprotokolle, die der Web Service anbietet. Diese Informationen sind maschinenlesbar.

Servicekonsumenten erstellen auf Basis der Informationen aus der WSDL-Datei einen SOAP-Request, um damit auf eine Operation zuzugreifen, die der Web Service bietet.

Der Web Service bearbeitet den SOAP-Request und sendet das Ergebnis als SOAP-Response an den Servicekonsumenten zurück.

Web Services über UDDI anbieten/abrufen

Alternativ, können Sie einen Web Service auch über ein Service-Verzeichnis (UDDI) anbieten/abrufen:



Ein Serviceanbieter veröffentlicht die Beschreibungen seines Web Services als WSDL-Datei in einer UDDI:

- Die WSDL-Datei beschreibt die Operationen, Daten und Austauschprotokolle, die der Web Service anbietet. Diese Informationen sind maschinenlesbar.
- Die UDDI fungiert als Servicebroker und ermöglicht es Servicekonsumenten, Web Services zu finden.

Ein Servicekonsument

1. durchsucht die UDDI und wählt den gewünschten Web Service aus.
2. wird dynamisch an den Serviceanbieter gebunden.
3. erstellt auf Basis der Informationen aus der WSDL-Datei einen SOAP-Request, um damit auf eine Operation zuzugreifen, die der Web Service bietet.

Der Web Service bearbeitet den SOAP-Request und sendet das Ergebnis als SOAP-Response an den Servicekonsumenten zurück.

40.2 Modulvariablen des Web Services Connectors

Web Services Input Listener Connector

Ein Web Services Input Listener Connector setzt folgende Modulvariablen:

Name	Erläuterung
`httpheader.request.SOAPAction`	Zweck des Requests
`httpheader.request.remoteAddress`	IP-Adresse des Aufrufers
`WSServerNameAndPort`	URL bzw. IP-Adresse und Port, unter der der Service aufgerufen wurde

Web Services Output Connector

Ein Web Services Output Connector setzt folgende Modulvariablen:

Name	Erläuterung
`ISHttpResponseCode`	HTTP-Antwortcode

Anhänge und Binärdaten behandeln

Ein Web Services Connector setzt bzw. liest folgende Modulvariablen, wenn eine der folgenden Optionen gesetzt ist:

- MTOM-Anhänge für ausgehende Nachricht aktivieren
- Binärdaten aus Antwort-Nachricht extrahieren und als Variablen-Inhalte setzen

Name	Erläuterung
`WSAttachmentFileName.<number>`	Dateinamen der einzelnen Anhänge
`WSAttachment.<number>`	Nummer des Anhangs
`WSAttachmentName.<number>`	Name des Anhangs

Diese Variablen werden sowohl für den Input- als auch für den Output-Konnektor gesetzt. Der Wert für `number` beginnt bei 0.

Web Services Input Connector

Ein Web Services Input Connector setzt folgende Modulvariablen, wenn die Option „Binärdaten aus Antwort-Nachricht extrahieren und als Variablen-Inhalte setzen“ markiert ist:

Name	Erläuterung
`WSAttachmentNumber`	Gesamtanzahl der Anhänge
`WSAttachmentContent-ID.<number>`	eindeutige ID des Anhangs
`WSAttachmentContent-Type.<number>`	Typ des Anhangs

Der Wert für `number` beginnt bei 0.

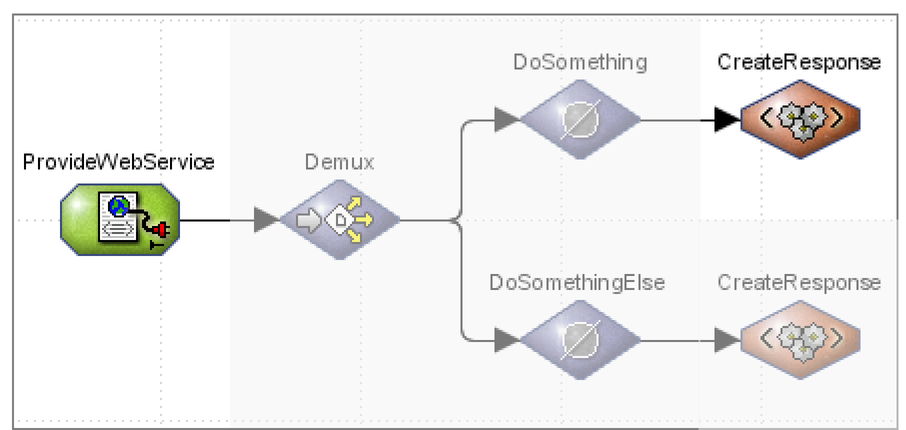
→ Siehe auch *Modulvariablen definieren (Workbench: Benutzer-Guide, Kap. 14, S. 369)*.

40.3 Web Service anbieten

Überblick

Ein Web Service-Provider wird in der inubit Suite 6 durch einen Technical Workflow mit einem Web Services Input Listener Connector realisiert. Der Web Services Input Listener Connector bietet die Funktionalität des Technical Workflows nach außen als Web Service an.

Das folgende Beispiel zeigt die wesentlichen Module für einen Web Service-Provider:



1. **Web Services Listener Connector „ProvideWebService“**

Der Web Services Input Listener Connector bietet u. a. die Operation `sayHello` an und wartet auf Aufrufe, die SOAP-Requests, von anderen Web Services.

Wenn ein SOAP-Request eintrifft, wird dieser an das folgende Modul, den Demultiplexer, weitergeleitet und damit der Workflow gestartet.

2. **Demultiplexer „Demux“ (optional)**

   Ein SOAP-Request kann genau eine Operation des Web Services aufrufen, auch wenn der Web Service mehrere Operationen anbietet. Um anhand der XML-Elemente im Request zu erkennen, welche Operation ausgeführt wird, kann ein Demultiplexer genutzt werden.

   Der Demultiplexer leitet den Request in den passenden Zweig des Workflows weiter, in denen der Request verarbeitet wird. In dem abgebildeten Beispiel sind die beiden Empty-Module „DoSomething“ und „DoSomethingElse“ Platzhalter für weitere verarbeitende Module.

3. **XSLT Converter „CreateResponse“**

   Das Ergebnis der Verarbeitung, die SOAP-Response, wird vom letzten Modul des Technical Workflows in einer Web Service-konformen Darstellung an den aufrufenden Web Service zurückgegeben. Diese SOAP-Response wird mit einem XSLT Converter erstellt.

So gehen Sie vor

1. **Web Services Input Listener Connector konfigurieren**

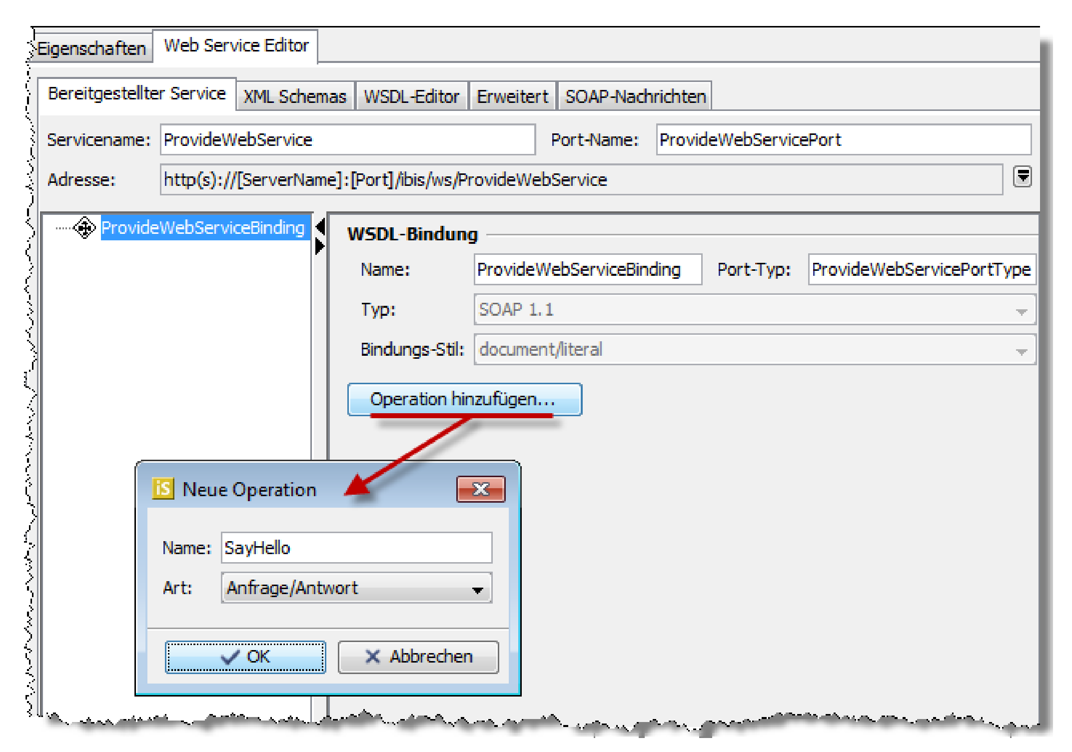
   a. Erzeugen Sie einen Technical Workflow.

   b. Fügen Sie einen Web Services Input Listener Connector ein. Legen Sie beim Konfigurieren fest, dass die Definition des aufzurufenden Web Services am Modul gespeichert werden soll.

      → Siehe *Dialog „Bereitgestellter Web Service“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.14.1, S. 436).*

   c. Öffnen Sie das Kontextmenü des Konnektors und wählen Sie „Bearbeiten“, um den Web Service-Editor anzuzeigen.

   d. Klicken Sie auf „Operation hinzufügen“. Der folgende Dialog öffnet sich:



e. Benennen Sie die Operation, z. B. mit `SayHello`.

f. Wählen Sie bei Art „Anfrage/Antwort“ aus. Damit legen Sie fest, dass Ihr Web Service eine SOAP-Response an den aufrufenden Web Service zurückgibt, statt nur dessen Request zu verarbeiten.

g. Klicken Sie auf „OK“, um den Dialog zu schließen.

   Beim Schließen des Dialog wird aus Ihren Angaben die WSDL Ihres Web Services erzeugt, in der u. a. die Strukturen der SOAP-Requests und - Responses definiert sind.

   Sie können die WSDL im Register „WSDL-Editor“ anzeigen und die SOAP-Requests/Responses im Register „SOAP-Nachrichten“.

h. Publizieren Sie den Konnektor.

2. **SOAP-Response erstellen**

   a. Erzeugen Sie einen XSLT Converter, fügen Sie diesen in den Technical Workflow ein und verbinden Sie ihn mit dem Web Services Input Listener Connector.

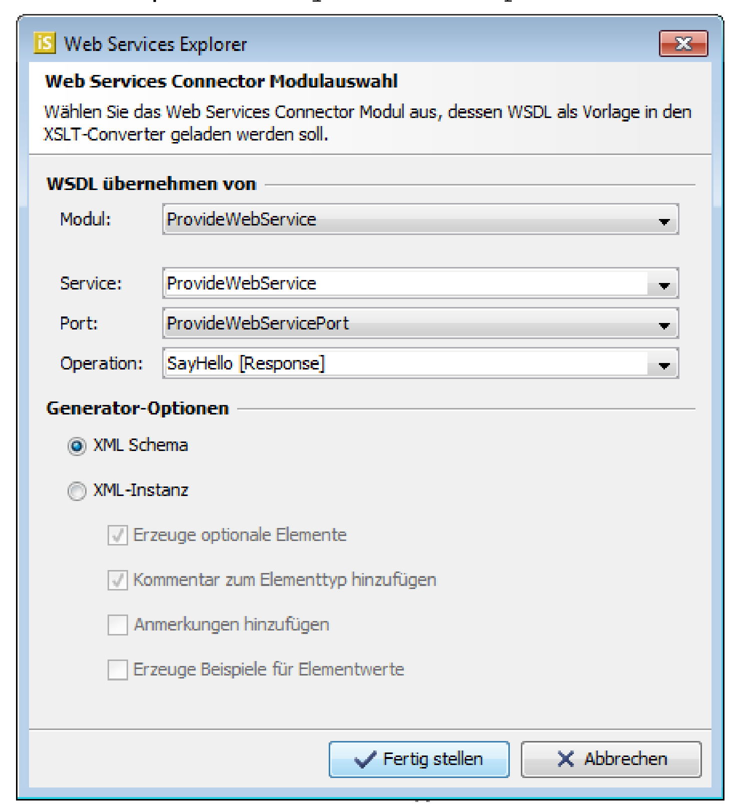
   b. Öffnen Sie den XSLT Converter zum Bearbeiten.

      Das Ergebnis der Workflow-Verarbeitung muss WSDL-konform als SOAP-Response strukturiert werden, bevor es an den aufrufenden Web Service zurückgesendet werden kann.

   c. In diesem Beispiel fand keine Verarbeitung statt, daher erstellen Sie nun lediglich ein Ausgangsmapping.

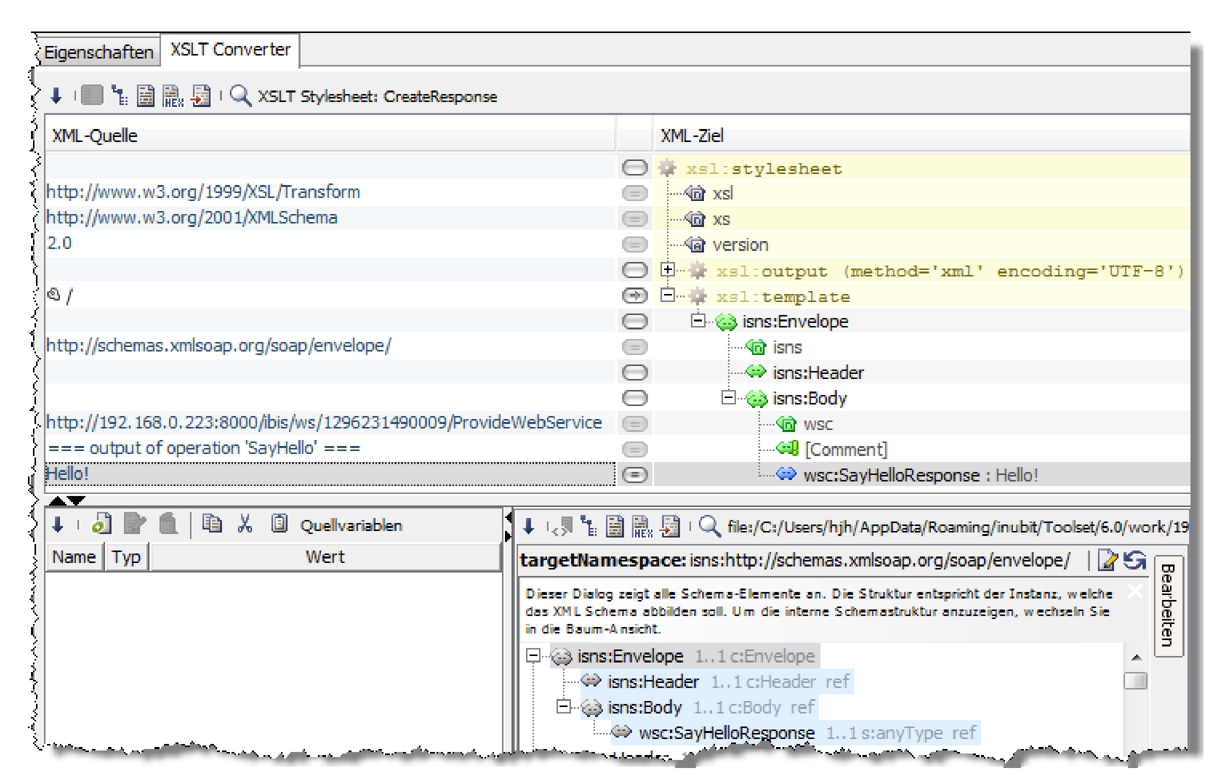
      Öffnen Sie im Bereich „XML-Zieldatei“ das ⬇ -Menü und wählen Sie „Öffnen von > Web Services Explorer“. Ein Dialog öffnet sich.

d. Wählen Sie bei „Modul“ Ihren Web Services Input Listener aus und bei „Operation“ `SayHello [Response]`:



e. Klicken Sie auf „Fertig stellen“. Die Struktur der SOAP-Nachricht wird angezeigt.

f. Ziehen Sie das Root-Element der erzeugten SOAP-Nachricht per Drag'n'Drop nach oben auf das `xsl:template`-Element und passen Sie z. B. den Wert des `sayHelloResponse`-Elements an:



g. Publizieren Sie den XSLT Converter.

3. **Testen**

   Erstellen Sie einen zweiten Web Service, der Ihren soeben erstellten Web Service aufruft.

   → Siehe *Web Service aufrufen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.4, S. 413).*

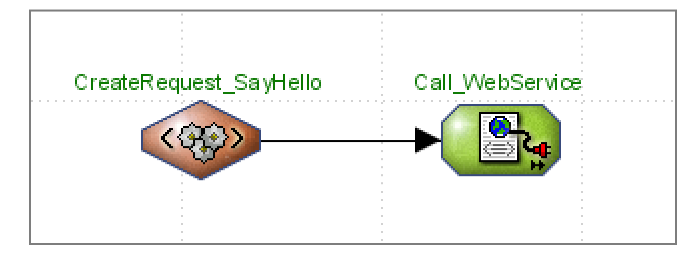
4. Publizieren Sie den Technical Workflow.

Sie können Ihren selbst erstellten Web Service anderen Benutzern über ein UDDI-Verzeichnis zugänglich machen.

→ Siehe *Web Service in UDDI publizieren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.10, S. 429).*

40.4 Web Service aufrufen

Überblick Das Beispiel zeigt die beiden wesentlichen Module für einen Web Service-Aufruf:



- **XSLT Converter** zum Erstellen der Web Service-Anfrage
- **Web Services Medium oder Output Connector** zum Senden der Anfrage und zum Empfangen des Ergebnisses (wenn vorhanden):
  - Ein **Medium Connector** kann eingesetzt werden, wenn der Web Service ein Ergebnis zurücksendet, das im Workflow weiterverarbeitet werden soll.

    Der Medium Connector sendet die Anfrage, wartet auf das Ergebnis und übergibt es an das nachfolgende Modul im Workflow. So kann das Ergebnis z. B. in eine Datenbank geschrieben oder für Drittsysteme geeignet aufbereitet werden.
  - Der Einsatz eines **Output Connectors** ist sinnvoll, wenn die Anfrage eine One-Way-Operation ohne Rückgabe aufruft und keine weiteren Prozessschritte nötig sind. Technisch verhält sich der Output Connector genauso wie der Medium Connector, d. h. auch der Output Connector wartet auf das Ergebnis und hält es vor.

So gehen Sie vor

1. **Web Services Connector konfigurieren**

   a. Erzeugen Sie einen Web Services Output Connector. Legen Sie beim Konfigurieren fest, dass die Definition des aufzurufenden Web Services am Modul gespeichert werden soll.

      → Siehe *Dialog „Aufzurufender Web Service“ (Workbench/ Process Engine: Systemkonnektor-Guide, Kap. 40.14.2, S. 438).*

   b. Geben Sie im Web Services Editor an, wo die WSDL des aufzurufenden Web Services zu finden ist und laden Sie diese WSDL.

Zum Testen können Sie z. B. den Web Service-Provider nutzen, dessen Erstellung im Abschnitt *Web Service anbieten (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.3, S. 409)* erläutert wird. Wenn dieser Web Service-Provider vorhanden und publiziert ist, gehen Sie so vor, um die Adresse der WSDL herauszufinden: Klicken Sie im Register „Aufzurufender Service“ am rechten Ende der Zeile auf den ⇟ Button und wählen Sie „Publizierte Services anzeigen“. Im Web-Browser öffnet sich eine Liste aller Web Services. Klicken Sie auf den Link „[NamedesWebServices]“. Die WSDL wird

angezeigt. Kopieren Sie die URL in die Zwischenablage und fügen Sie diese im Register „Aufzurufender Service“ in das Feld „URL“ ein.

---

c. Publizieren Sie den Konnektor.

d. Erzeugen Sie einen Technical Workflow und fügen Sie den Web Services Output Connector ein.

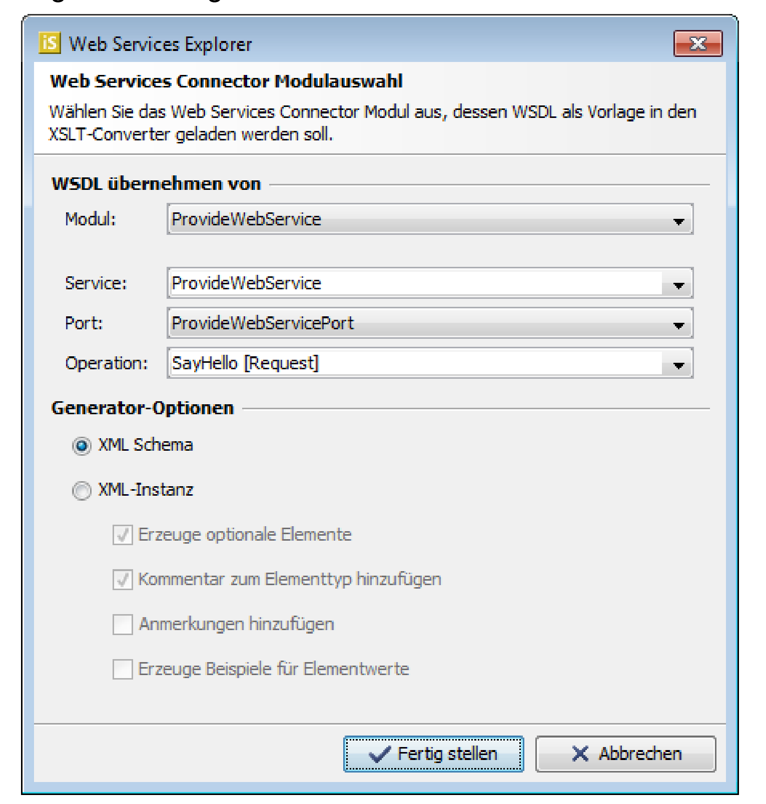
2. **SOAP-Request erstellen**

a. Erzeugen Sie einen XSLT Converter und markieren Sie beim Konfigurieren die Option „Eingangsnachricht ignorieren“.

b. Fügen Sie den XSLT Converter in den Technical Workflow ein.

c. Öffnen Sie den XSLT Converter zum Bearbeiten.

d. Öffnen Sie im Bereich „XML-Zieldatei“ das -Menü und wählen Sie „Öffnen von > Web Services Explorer“. Der folgende Dialog öffnet sich:

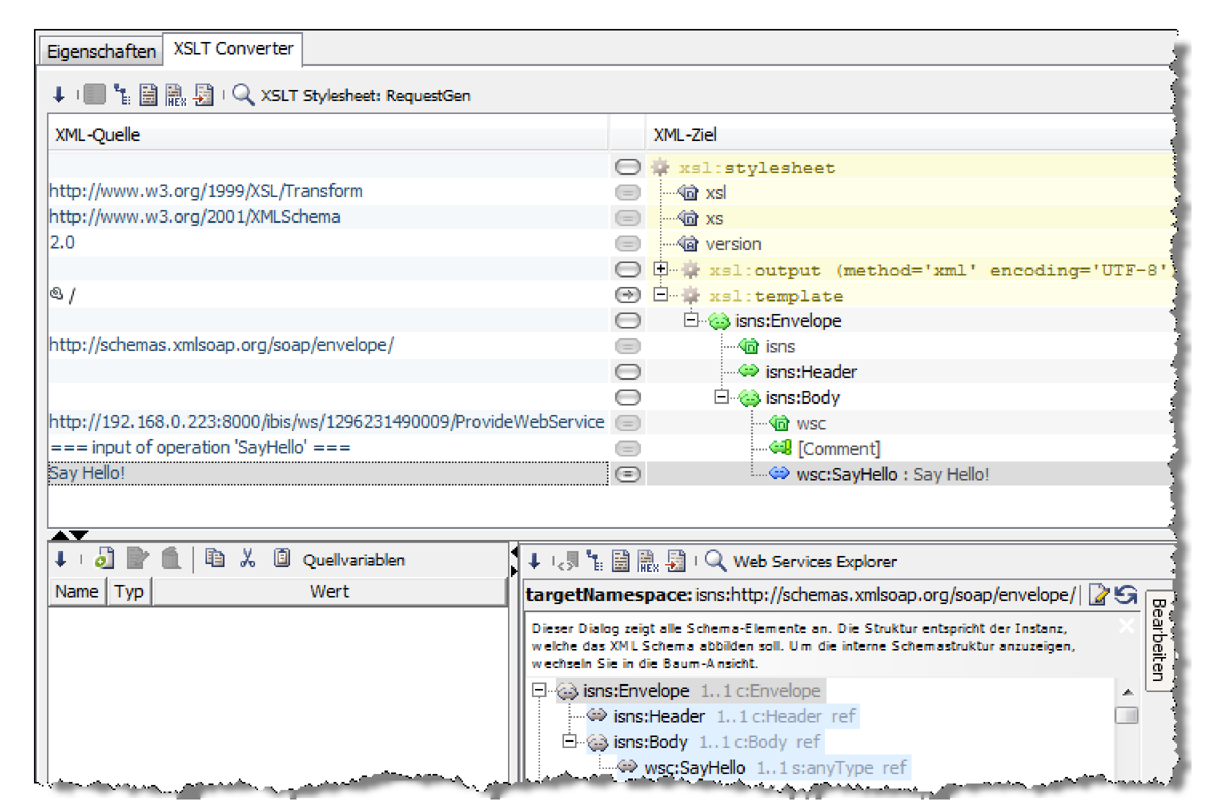


e. Wählen Sie den Web Services Connector aus und die aufzurufende Operation (in Zusammenhang mit „Request"). Klicken Sie auf „Fertig stellen“, um den Dialog zu schließen.

Die Struktur des SOAP-Requests, die der Web Service erwartet, wird angezeigt.

f. Ziehen Sie das Root-Element der erzeugten SOAP-Nachricht per Drag'n'Drop nach oben auf das `xsl:template`-Element.

g. Passen Sie die Werte im Bereich „XML-Quelle“ an, z. B.:



h. Publizieren Sie den XSLT Converter.

3. **Testen**

a. Setzen Sie im Technical Workflow einen Startpunkt an den XSLT Converter und wählen Sie im Kontextmenü „Test ohne Datei starten".

b. Öffnen Sie nach dem Test den Watchpoint hinter dem Web Services Connector, um das Ergebnis, die SOAP-Response, anzuzeigen.

Um dynamische Web Service-Aufrufe zu erzeugen, fügen Sie vor dem XSLT Converter weitere Module ein, die z. B. die zu verwendenden Web Service-Aufrufparameter aus einer Datenbank auslesen und an den XSLT Converter übergeben. Im XSLT Converter bilden Sie diese Parameter dann auf Teile der SOAP-Nachricht ab.

40.5 Asynchrone Web Services

Dieser Abschnitt erläutert die folgenden Themen:

- *Asynchronen Web Service anbieten, S. 418*
- *Asynchronen Web Service aufrufen, S. 420*

Verwendung

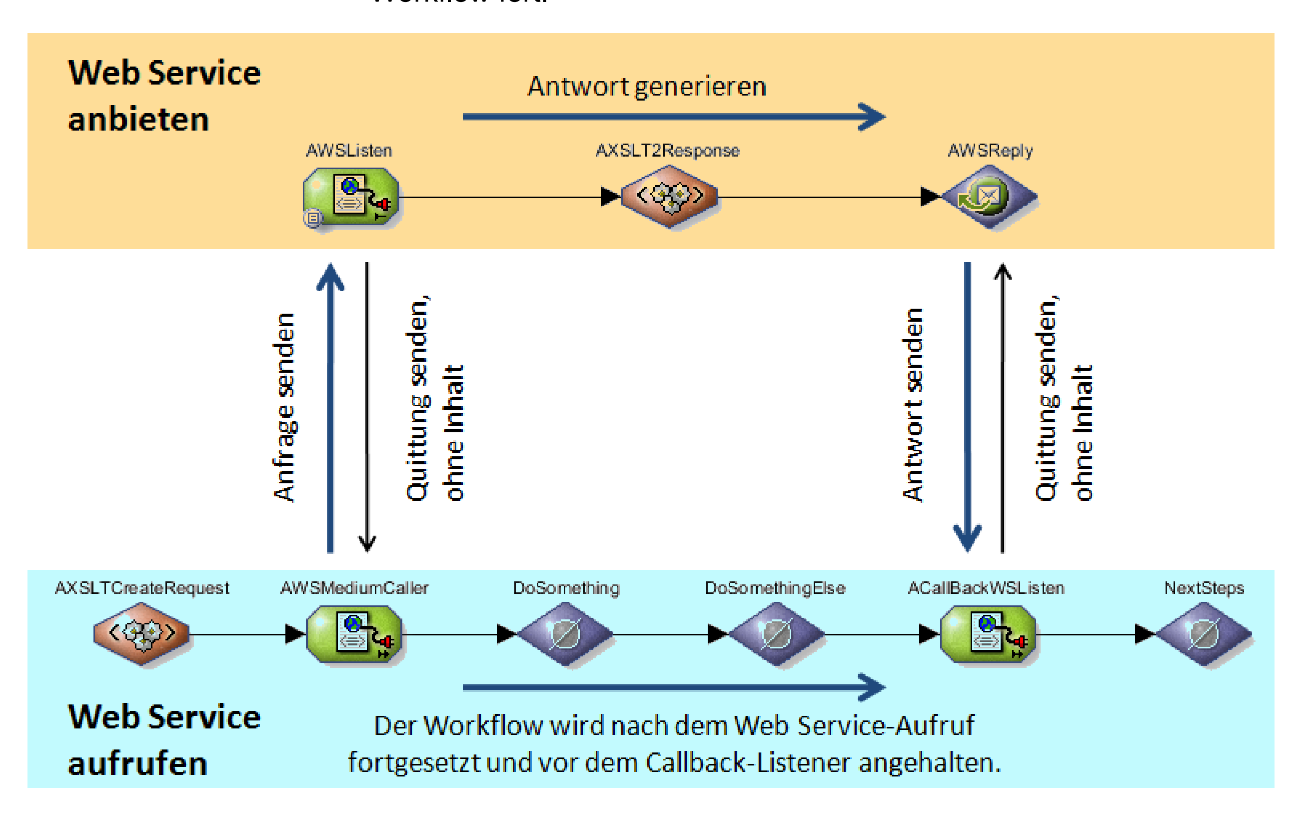
Bei einem asynchronen Web Service sind das Versenden der Anfragenachricht und das Erhalten der Antwortnachricht entkoppelt. Beide Nachrichten werden über separate HTTP-Verbindungen gesendet. Damit kann die Bearbeitung der Anfrage länger dauern, ohne dass die Verbindung wegen eines Timeouts unterbrochen wird.

Asynchrone Web Services sind daher insbesondere für langlaufende Services geeignet.

Im Folgenden wird der Kommunikationsablauf bei einem asynchronen Service-Aufruf skizziert. In dem Beispiel sind beide Kommunikationspartner über Technical Workflows implementiert. In der Praxis muss dies natürlich nicht der Fall sein.

Funktionsprinzip

Ein Workflow, der einen asynchronen Web Service anfordert, wird bis zu einem Callback Listener abgearbeitet. Der Workflow des aufgerufenen Web Services generiert währenddessen die Antwort. Ein Reply-Modul am Ende des aufgerufenen Workflows sendet die Antwort an den aufrufenden Workflow. Der Callback Listener im aufrufenden Workflow nimmt die Antwort entgegen und setzt den Workflow fort.



Voraussetzungen

- Um einen asynchronen Web Service mit einem beliebigen Web Service-Client aufzurufen, muss der Client WS-Addressing unterstützen und eine nicht-anonyme Antwort-Adresse (wsa:ReplyTo) verwenden.
- Ein asynchroner Web Service, der eine nicht-anonyme Antwort-Adresse (wsa:ReplyTo) erwartet, muss über einen Medium Web Service Connector aufgerufen werden.

40.5.1 Asynchronen Web Service anbieten

Ein Workflow, der einen asynchronen Web Service anbieten soll, muss mindestens aus folgenden Modulen bestehen:

- Asynchroner Web Service Listener
- XSLT Converter zur Erstellen der Antwort
- Reply-Modul zum Senden der Antwort an den Callback Listener

Asynchronen Web Service Listener anlegen

Dieser Listener nimmt die Anfragen entgegen, bearbeitet sie und leitet sie an das nächste Modul im Workflow weiter.

So gehen Sie vor

1. Legen Sie in einem neuen Workflow einen neuen Web Services Connector an, benennen Sie ihn und klicken Sie auf „Weiter“, um zum Register „System Connector Eigenschaften“ zu gelangen.
2. Wählen Sie auf dem Register „System Connector Eigenschaften“ für „Verwendung“ den Eintrag „Web Service Aufrufe empfangen (Web Service bereitstellen)“.
3. Wählen Sie „Aktiv“ als „Connector Status“ und klicken Sie auf „Weiter“, um zum Register „Bereitgestellter Web Service“ zu gelangen.
4. Wählen Sie eine WSDL-Bindung bzw. „neu“, den Typ und den Bindungs-Stil.
5. Passen Sie ggf. den Namensraum und die Bindungseinstellungen an.

   → Siehe

   - *Operationen eines Web Services erstellen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.8, S. 426)*
   - *Dialog „Bereitgestellter Web Service“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.14.1, S. 436)*
6. Klicken Sie auf „Weiter“, um zum Register „Modul-Editor“ zu gelangen.

7. Wechseln Sie ggf. auf das Register „Bereitgestellter Web Service“ und klicken Sie auf den Button „Operation hinzufügen“.
8. Benennen Sie die Operation (z. B. „SayHello“) und wählen Sie ggf. als Art „Anfrage/Antwort“.
9. Klicken Sie auf „OK“.
10. Aktivieren Sie auf dem Register „Erweitert“ die Checkbox „Unterstützung für asynchronen Antwort-Versand“.

SOAP-Nachricht speichern

So gehen Sie vor

1. Wechseln Sie auf das Register „SOAP-Nachricht“.
2. Wählen Sie die beim Konfigurieren des asynchronen Web Service Listeners definierte Operation (z. B. „SayHello“) und die dafür generierte Eingangsnachricht.
3. Klicken Sie auf „Anzeigen“, damit die Nachricht in den Editor geladen wird.
4. Speichern Sie die SOAP-Eingabenachricht als Datei in Ihr Dateisystem.
5. Klicken Sie auf „Fertigstellen“.
6. Publizieren Sie das Modul.

XSLT Converter zum Erstellen der Antwort konfigurieren

Dieser XSLT Converter erstellt aus der Ausgangsnachricht des Web Services eine Antwortnachricht und übergibt sie an das Reply-Modul.

So gehen Sie vor

1. Legen Sie einen neuen XSLT Converter an und benennen Sie ihn.
2. Klicken Sie zwei Mal auf „Weiter“, um zum Modul-Editor zu gelangen.

   Auf dem Register „XSLT Converter Eigenschaften“ sind keine Anpassungen erforderlich.
3. Öffnen Sie im Fensterbereich „XML-Zieldatei“ einen Web Services Explorer.
4. Wählen Sie als Modul den oben definierten Web Services Listener und als Operation die Response-Operation.
5. Ziehen Sie das Root-Element der erzeugten Nachricht aus dem Bereich „targetNamespace“ (in der Schema-Ansicht) auf das Element „xsl:template“ im Bereich „XML-Ziel“ rechts oben.
6. Tragen Sie im Bereich „XML-Quelle“ (links oben) den Text der Response-Nachricht ein.
7. Publizieren Sie das Modul.
8. Ziehen Sie eine Verbindung vom Web Services Listener zum soeben angelegten XSLT Converter.

Reply-Modul zum Senden der Antwort konfigurieren

Das Reply-Modul sendet die vom XSLT Converter generierte Antwort-Nachricht an den Web Service-Aufrufer/Client.

Wenn Sie den Web Service-Aufrufer über einen Technical Workflow implementiert haben, dann ist dies der Callback Listener des aufrufenden Workflows.

Alternativ zu einem Reply-Modul können Sie einen Medium/Output Web Services Connector so konfigurieren, dass er die asynchrone Antwort eines Web Services an einen Callback Listener sendet. Weitere Informationen finden Sie im Abschnitt *Register „Aufzurufender Service“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.14.3.2, S. 440)*.

So gehen Sie vor

1. Legen Sie ein neues Reply-Modul an und benennen Sie es.
2. Lassen Sie alle voreingestellten Werte unverändert.
3. Klicken Sie auf „Fertigstellen“.
4. Publizieren Sie das Modul.
5. Ziehen Sie eine Verbindung vom XSLT Converter zum Reply-Modul.

Wenn Sie alle drei Module konfiguriert und verbunden haben, publizieren Sie den Workflow.

Als nächstes erstellen Sie einen Workflow zum Aufrufen eines asynchronen Web Services.

→ Siehe *Asynchronen Web Service aufrufen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.5.2, S. 420).*

40.5.2 Asynchronen Web Service aufrufen

Ein Workflow, der einen asynchronen Web Service anfordern soll, muss mindestens aus folgenden Modulen bestehen:

- Modul zum Erstellen des Aufrufs, z. B. ein XSLT Converter oder das Variablen-Mapping
- Asynchroner Web Service Medium Connector zum Senden des Aufrufs
- Web Services Callback Listener zum asynchronen Entgegennehmen der vom aufgerufenen Web Services gesendeten Antwort.

Zwischen dem aufrufenden Web Services Connector und dem Web Services Callback Listener können Sie beliebige Module einfügen.

Die Module des aufrufenden und des aufgerufenen Web Services müssen Sie in zwei verschiedenen Workflows anlegen.

XSLT Converter zum Erstellen einer Anfrage konfigurieren

Dieser XSLT Converter erstellt Anfragen an einen Web Service.

So gehen Sie vor

1. Legen Sie in einem neuen Workflow einen neuen XSLT Converter an und benennen Sie ihn.
2. Klicken Sie auf „Weiter“, um zum Register „XSLT Converter Eigenschaften“ zu gelangen.
3. Aktivieren Sie die Checkbox „Eingangsnachricht ignorieren“.
4. Klicken Sie auf „Weiter“, um zum Modul-Editor zu gelangen.
5. Öffnen Sie im Fensterbereich „XML-Zieldatei“ einen Web Services Explorer.
6. Wählen Sie als Modul den oben definierten Web Services Listener und als Operation die Request-Operation.

   → Siehe *Asynchronen Web Service anbieten (Workbench/ Process Engine: Systemkonnektor-Guide, Kap. 40.5.1, S. 418).*
7. Ziehen Sie das Root-Element der eingelesenen Nachricht aus dem Bereich „targetNamespace“ (in der Schema-Ansicht) auf das Element „xsl:template“ im Bereich „XML-Ziel“ rechts oben.
8. Tragen Sie in der Zeile mit dem XML-Ziel-Element „wsc:<Name Ihrer Operation>“ im Bereich „XML-Quelle“ (links oben) den Text der Request-Nachricht ein.
9. Publizieren Sie das Modul.

Asynchronen Web Service Medium Connector anlegen

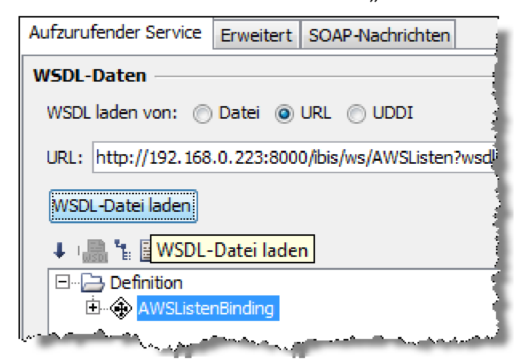
Dieser Web Service Medium Connector sendet die vom soeben angelegten XSLT Converter generierten Aufrufe an einen Web Service.

So gehen Sie vor

1. Legen Sie einen neuen Web Services Connector an, benennen Sie ihn und klicken Sie auf „Weiter“, um zum nächsten Register zu gelangen.
2. Wählen Sie auf dem Register „System Connector Eigenschaften“ für „Verwendung“ den Eintrag „Web Service aufrufen“.
3. Wählen Sie „Aktiv“ als „Connector Status“ und klicken Sie zwei Mal auf „Weiter“, um zum Register „Modul-Editor“ zu gelangen.

   → Siehe *Dialog „Aufzurufender Web Service“ (Workbench/ Process Engine: Systemkonnektor-Guide, Kap. 40.14.2, S. 438).*
4. Geben Sie im Bereich „WSDL-Daten“ die URL zur WSDL des aufzurufenden Services an.

a. Klicken Sie im Register „Aufzurufender Service“ am rechten Ende der Zeile auf das Icon ▾ und wählen Sie „Publizierte Services anzeigen“. Im Web-Browser öffnet sich eine Liste aller Web Services.
b. Klicken Sie auf den Link mit dem Namen des gewünschten Web Services. Die WSDL wird angezeigt.
c. Kopieren Sie die URL in die Zwischenablage.
d. Fügen Sie die URL im Register „Aufzurufender Service“ in das Feld „URL“ ein.
e. Klicken Sie auf den Button „WSDL-Datei laden“.



Die Checkbox „Antwort asynchron empfangen / separate Antwort-Adresse mitschicken“ wird automatisch aktiviert. Das Feld „Antwort-URL (wsa:ReplyTo)“ wird automatisch auf die korrekten Werte gesetzt. Ausschlaggebend dafür sind Policies in der WSDL.

5. Klicken Sie auf „Fertigstellen“.
6. Publizieren Sie das Modul.
7. Ziehen Sie eine Verbindung vom Request XSLT Converter zum soeben konfigurierten Web Services Medium Connector.

Weitere Module anlegen

Während der Web Service die Anfrage bearbeitet und die Antwort erstellt, wird der aufrufende Workflow fortgesetzt. Der Workflow stoppt unmittelbar vor dem Callback Listener.

So gehen Sie vor

1. Legen Sie mehrere Module gemäß Ihren Anforderungen an.
2. Verknüpfen Sie die Module untereinander und mit dem aufrufenden Web Services Connector.

Callback Listener Connector anlegen

Dieser Web Services Medium Listener Connector wartet auf die Antwort des Web Services. Der Workflow stoppt unmittelbar vor diesem Connector und wird erst fortgesetzt, nachdem die Antwort eingetroffen ist.

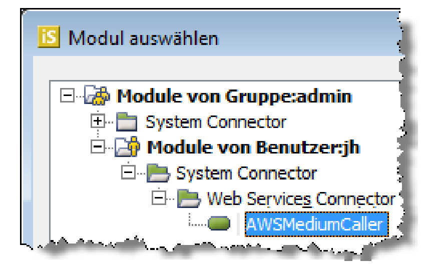
Liegt die Antwort bereits vor, bevor der Workflow den Callback Listener erreicht hat, wird sie zwischengespeichert.

So gehen Sie vor

1. Legen Sie einen neuen Web Services Connector an, benennen Sie ihn und klicken Sie auf „Weiter“, um zum nächsten Register zu gelangen.
2. Wählen Sie auf dem Register „System Connector Eigenschaften“ für „Verwendung“ den Eintrag „Asynchrone Antwort-Nachrichten (Callbacks) empfangen“.

Die Option „Medium Connector (Workflow-Mitte)“ und die Checkbox „Auf Datenempfang warten (Listener Connector)“ werden automatisch aktiviert.

3. Klicken Sie auf „Weiter“, um zum Register „Bereitgestellter Web Service“ zu gelangen.
4. Klicken Sie im Bereich „Callback-Listener spezifische Einstellungen“ in der Zeile „WSDL von zugehörigem Aufrufer-Modul importieren“ auf das Icon [...].
5. Wählen Sie den aufrufenden Web Services Connector, den Sie am Anfang dieses Workflows angelegt haben.



Die weiteren Angaben auf dem Register „Bereitgestellter Web Service“ können Sie nicht ändern, da sie automatisch korrekt voreingestellt werden.

6. Klicken Sie auf „Weiter“, um das Register „Modul-Editor“ anzuzeigen.
7. Zeigen Sie das Register „Erweitert“ an.
8. Aktivieren Sie im Bereich „Callback-Listener spezifische Einstellungen“ die Checkbox „Listener nimmt Antwort-Nachrichten entgegen (anstelle von Anfrage-Nachrichten)“, sofern sie nicht bereits automatisch aktiviert wurde.
9. Klicken Sie auf „Fertigstellen“.
10. Verbinden Sie das letzte der weiter oben konfigurierten „weiteren Module“ mit dem soeben angelegten Callback Listener Connector.

Wenn Sie alle Module konfiguriert und verbunden haben, dann publizieren Sie den Workflow.

40.6 Web Services Provider durch Security Token Service absichern

Dieser Abschnitt erläutert, wie Sie einen Web Service Provider über einen Security Token Service (STS) absichern.

Voraussetzungen

- Der Web Service Provider, der gesichert werden soll, wurde mit der inubit Suite 6 erstellt. Die URL des STS ist bekannt.
- Folgende Daten liegen vor:
  - Keystore mit dem privaten Schlüssel des zu sichernden Web Services.
  - Truststore mit dem öffentlichen Schlüssel des STS.

So gehen Sie vor

1. Öffnen Sie den Web Services Connector zum Bearbeiten.
2. Aktivieren Sie im Register „Erweitert“ im Bereich „W3C-Standards“ die Option „WS-Security“ und klicken Sie auf den Button „Einstellungen“.

   Der Dialog „WS-Security Konfiguration“ öffnet sich.
3. Geben Sie im Bereich „Service-Authentifizierung“ das Keystore-Passwort an und importieren Sie den Keystore.
4. Wählen Sie im Bereich „Consumer-Authentifizierung“ unter „Security-Mechanismus“ die Option „Security Token Service issued Token with service certificate (STS)“ aus.
5. Geben Sie im Bereich „Consumer-Authentifizierung“ im Feld „STS Adresse“ die URL des STS an, der Ihren Web Service absichern soll.
6. Geben Sie an, wie die Kommunikation Ihres Web Service Providers mit dem STS gesichert werden soll:

   Wählen Sie „X.509“ und importieren Sie den Truststore mit dem öffentlichen Schlüssel des STS.
7. Klicken Sie auf „Fertig stellen“.
8. Publizieren und aktivieren Sie den Workflow.

In der WSDL-Security Policy des Web Service Providers ist nun beschrieben, wie und an welchem STS Consumer sich authentifizieren müssen.

Falls noch nicht geschehen, registrieren Sie den Web Service Provider am STS. Hinweise zum Registrieren an einem inubit Suite 6-internen STS finden Sie im Abschnitt *Web Services Provider an STS Connector registrieren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 33.2, S. 350)*.

40.7 STS-gesicherten Web Service Provider aufrufen

Dieser Abschnitt erläutert, wie Sie einen Web Service Consumer konfigurieren, damit dieser einen STS-gesicherten Web Service Provider aufrufen kann.

→ Siehe *Funktionsprinzip des STS Connectors (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 33.1, S. 348).*

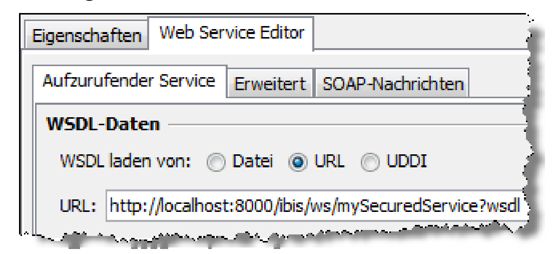
Voraussetzungen

- Sie benötigen einen Truststore mit dem öffentlichen Schlüssel des STS oder dessen Zertifikat.
- Web Services Provider, Web Service Consumer und STS sind in der inubit Suite 6 realisiert.

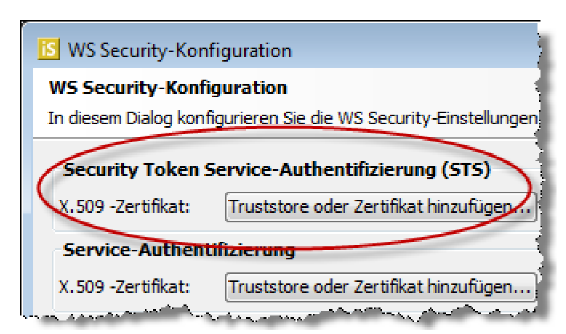
Um mit einem externen Consumer einen Web Service aufzurufen, der über einen STS der inubit Suite 6 abgesichert ist, konfigurieren Sie dessen Sicherheitseinstellungen entsprechend den Angaben des Herstellers.

So gehen Sie vor

1. Öffnen Sie den Web Services Connector Ihres Web Services Consumers zum Bearbeiten.
2. Geben Sie im Register „Aufzurufender Service“ die URL an, unter der die WSDL des STS-gesicherten Web Service Providers verfügbar ist:

3. Laden Sie die WSDL.
4. Klicken Sie im Register „Erweitert“ im Bereich „W3C Standards“ auf den Button „WS-Security“. Der Dialog „WS-Security Konfiguration“ öffnet sich.
5. Importieren Sie den Truststore des STS oder dessen Zertifikat. Der Truststore mit dem öffentlichen Schlüssel des STS wird benötigt, um das vom STS signierte Token lesen zu können. Mit der Signatur versichert der STS, dass das Token vom STS selbst ausgestellt wurde.



Nach dem erfolgreichen Import wird die Gültigkeit des Schlüssels angezeigt.

6. Um die Kommunikation mit dem STS zu sichern, legen Sie im Bereich „Consumer-Authentifizierung“ fest, wie Ihr Consumer sich gegenüber den STS authentifizieren soll.

   Der Consumer sendet diese Benutzer/Passwort-Kombination als Teil seines Security Token Requests an den STS.
7. Klicken Sie auf „Fertig stellen“.
8. Publizieren und aktivieren Sie den Workflow

40.8 Operationen eines Web Services erstellen

Voraussetzungen

- Web Services Input Listener Connector, bei dem die WSDL-Datei am Modul selbst gespeichert ist
- Vorlagen (optional):

  XML Schema-Datei oder WSDL-Datei mit Operationen, welche den zu erstellenden ähnlich sind.

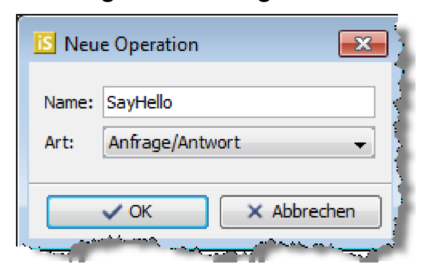
Die WSDL-Datei muss dieselbe Bindungsart unterstützen, die Sie bei der Konfiguration des Connectors im *Dialog „Bereitgestellter Web Service“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.14.1, S. 436)* angegeben haben.

So gehen Sie vor

1. Zeigen Sie im Web Service Editor das Register „Bereitgestellter Service“ an.
2. **Operation hinzufügen**

   Führen Sie eine der folgenden Aktionen durch:
   - Klicken Sie auf den Button „Operation hinzufügen“.
   - Markieren Sie das Wurzelelement, öffnen Sie das Kontextmenü und wählen Sie „Operation hinzufügen“.

Der folgende Dialog öffnet sich:



a. Geben Sie einen sprechenden Namen für die Operation ein.

b. Wählen Sie die Art der Operation aus.

- **Anfrage/Antwort**: Elemente für die Eingabe und die Rückgabe der Operation werden eingefügt.
- **Anfrage**: Es wird nur ein Element für die Eingabe eingefügt.

c. Schließen Sie den Dialog mit „OK“.

Die Operation wird mit dem angegebenen Namen und den ausgewählten Eingabe-/Rückgabe-Parametern unter dem Wurzelelement angezeigt.

3. **Nachrichtenteil einfügen**

- **Bei Bindungs-Stil „document/literal“**:

  Beim Anlegen der Operation wurden automatisch Nachrichten-Teile für die Eingabe- und Ausgabe-Nachricht hinzugefügt.

  Wenn nötig, können Sie den Typ des Nachrichten-Teils anpassen.

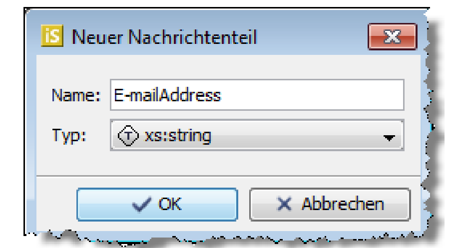
- **Bei Bindungs-Stil „RPC/encoded“ und „RPC/literal“**:

  Sie müssen Nachrichten-Teile für die Ein- und Rückgabe-Nachrichten definieren.

So gehen Sie vor:

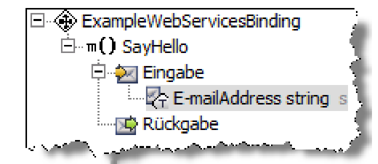
a. Markieren Sie in der Baumansicht den Parameter, dem Sie einen Nachrichtenteil hinzufügen möchten.

b. Öffnen Sie das Kontextmenü und wählen Sie „Nachrichten-Teil hinzufügen“. Der folgende Dialog öffnet sich:

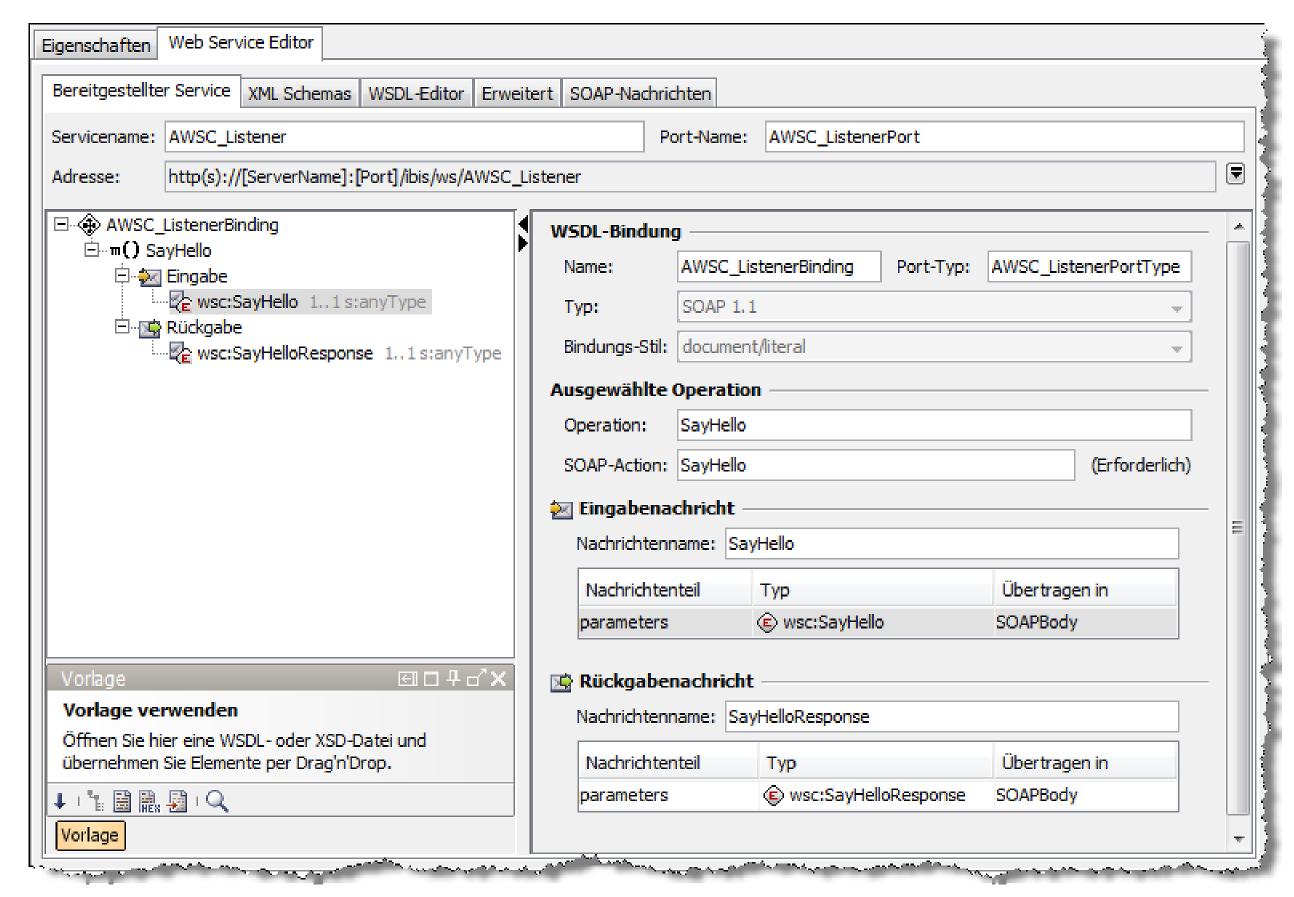


c. Geben Sie den Namen und Typ ein.

d. Schließen Sie den Dialog. Der Nachrichten-Teil wird unterhalb des dazugehörigen Parameters angezeigt, z. B.:



4. **Vorlage verwenden** (optional)
   Zum Typisieren der Parameter können Sie einfache oder komplexe Datentypen aus einer existierenden Schema-Datei per Drag'n'Drop verwenden. Auch die Operation können Sie aus anderen WSDL-Dateien per Drag'n'Drop übernehmen.
   Klicken Sie auf „Vorlage" am unteren Rand des Editors.
   Die Vorlagen-Palette öffnet sich.



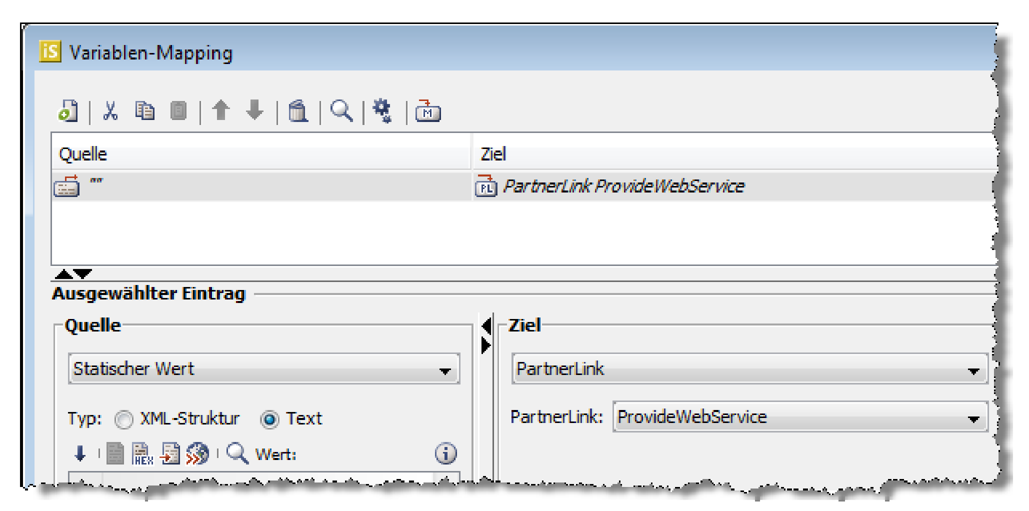
Fügen Sie entsprechend der Funktionalität, die der Web Service zur Verfügung stellen soll, weitere Operationen und Nachrichtenteile hinzu.

40.9 PartnerLinks überschreiben

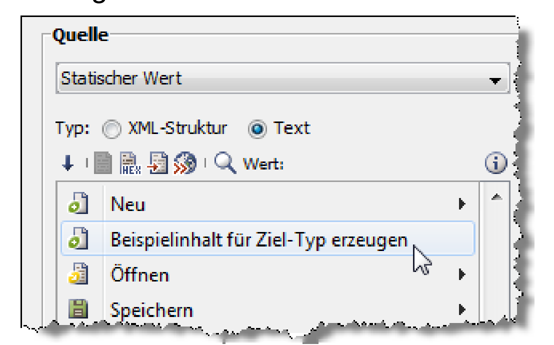
Sie können die Adresse des aufzurufenden Web Services mit Hilfe des Variablen-Mappings dynamisch zur Ausführungszeit ändern.

So gehen Sie vor

1. Öffnen Sie den Variablen-Mapping-Dialog am Web Services Connector.
2. Wählen Sie als Ziel den PartnerLink aus, der die Daten des aufzurufenden Web Services enthält.
3. Wählen Sie als Quelle z. B. „Statischer Wert":



4. Klicken Sie auf ⬇ und wählen Sie „Beispiel-Inhalt für Ziel-Typ erzeugen":



5. Ändern Sie die Adresse in dem `wsa:Address`-Element.

Die WSDL des Endpunkts wird als Metadatum in die XML-Struktur unterhalb von `wsa:Metadata` eingefügt. Falls die WSDL nicht geändert werden soll, können Sie das `wsdl:definitions`-Element entfernen.

40.10 Web Service in UDDI publizieren

Sie können Ihren Web Service in einem UDDI-Register publizieren. Damit stellen Sie den Web Service allen Benutzern zur Verfügung, die auf das gewählte UDDI-Register zugreifen können.

Voraussetzungen

Die UDDI muss installiert sein.

→ Siehe *Komponenten der inubit Suite 6 nachinstallieren (Process Engine: Administrator- und Entwickler-Guide, Kap. 3.3, S. 51).*

So gehen Sie vor

1. Zeigen Sie im Modul-Editor den Web Services Input Listener an.
2. Zeigen Sie das Register „Erweitert“ an.
3. Klicken Sie im Bereich „UDDI“ auf den Button „Publizieren“. Der *Dialog „UDDI Dateneinstellungen“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.14.6, S. 451)* wird angezeigt.
4. Füllen Sie den Dialog aus.
5. Klicken Sie auf „Publizieren“.

Der Web Service wird in dem gewählten UDDI-Register publiziert.

40.11 Aktive Web Services anzeigen

Eine Liste aller aktiven Web Services erhalten Sie im Web-Browser unter folgenden URL:

- inubit IS 4.1: http://myServer:8000/ibis/services
- inubit BPM-Suite 5.0: http://myServer:8000/ibis/services/listServices
- inubit BPM-Suite 5.3: http://myServer:8000/ibis/listServices
- inubit Suite 6: http://myServer:8000/ibis/listServices

Web Services, die mit Web Services Input Konnektoren realisiert sind, werden erst als aktiv gelistet, wenn die entsprechenden Technical Workflows aktiviert sind.

40.12 SOAP-Nachrichten protokollieren

Zur Fehleranalyse können Sie SOAP-Nachrichten der Web Services protokollieren lassen.

Die Protokolle finden Sie im Log-Verzeichnis Ihres Applikationsservers:

- **Tomcat**: `<iS-installdir>/server/Tomcat/logs/catalina.[Datum].log`, unter Linux zusätzlich `catalina.out`
- **JBoss**: `<iS-installdir>/server/JBoss/server/default/log/server.log` bzw. `server.log.[Datum]`

Das Logging steht nur für Web Service-Konnektoren ab Version 5.1 zur Verfügung, wenn diese das Metro-Framework nutzen. Das Logging kann also nicht für Web Service-Konnektoren im Kompatibilitätsmodus mit dem Axis2-Framework genutzt werden.

So gehen Sie vor

1. Erstellen Sie ein neues Java System Property `com.sun.xml.ws.transport.http.client.HttpTransportPipe.dump` mit dem Wert „true".
   → Siehe *Java System Properties (Process Engine: Administrator- und Entwickler-Guide, Kap. 2.17, S. 38).*
2. Speichern Sie Ihre Änderung.

40.13 Binärdaten als Attachments mit MTOM übertragen

Dieser Abschnitt erläutert die folgenden Themen:

Die Performance von Web Services wird durch die Übertragung großer Mengen von Binärdaten beeinträchtigt. MTOM (Message Transmission Optimization Mechanism) ist ein Mechanismus, um die Performance bei der Übertragung binärer Daten in Web Services zu optimieren. Dabei werden die Binärdaten nicht base64-kodiert in der SOAP-Nachricht, sondern als Anhang der SOAP-Nachricht verschickt.

Der vom W3C empfohlene MTOM-Standard spezifiziert, dass die Binärdaten als Inhalt eines Elements vom Typ `xsd:base64binary` enthalten sein müssen.

Siehe http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/.

40.13.1 Nachrichten mit binären Attachments versenden

Sie können Binärdaten mit MTOM als Anhänge von SOAP-Nachrichten versenden:

- Bei einem Medium/Output Connector erfolgt der Versand innerhalb der ausgehenden Nachricht.
- Bei einem Input Listener Connector erfolgt der Versand innerhalb der Response als Ausgangsnachricht.

Bei einem Input Listener Connector, der mit dem Metro-Framework, d. h. nicht im Kompatibilitätsmodus, betrieben wird, wird die Antwort nur dann MTOM-kodiert, wenn der Web Service Consumer seine MTOM-Unterstützung explizit signalisiert hat!

Der Web Service Consumer kann seine Unterstützung durch das Senden eines MTOM-Requests signalisieren oder indem im „Accept“-HTTP-Header der Eintrag „application/xop+xml“ hinzugefügt wird.
Falls Sie den Input Listener mit einem Medium/Output Connector der inubit Suite 6 aufrufen und eine MTOM-kodierte Antwort erwarten, müssen Sie bei dem Medium/Output Connector die Option „MTOM Anhänge für Antwort-Nachricht aktivieren“ markieren. Die Option finden Sie *Register „Erweitert“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.14.3.5, S. 442)*.

Wenn der Web Service Consumer keine explizite MTOM-Unterstützung signalisiert hat, dann werden die Anhangdaten inline in der SOAP-Nachricht gesendet.

Sie definieren jeweils die eigentlichen Daten (z. B. Grafiken als `jpeg` oder Dokumente als `pdf`) als Workflow-Variablen und passen die SOAP-Nachricht so an, dass ein Element vom Typ `xmime:base64Binary` mit dem Attribut `xmime:contentType` und dem jeweiligen Wert (z. B. image/jpeg) enthalten ist. Dabei müssen Sie auch eine Variable anlegen, die das Element der SOAP-Nachricht angibt, für das die Daten bestimmt sind.

Die Elemente der SOAP-Nachricht, deren Inhalt mit Hilfe von MTOM verschickt werden soll, müssen mit dem Attribut `xmime:contentType` belegt sein.

Voraussetzungen

- Web Services Medium/Output oder Input Listener Connector ist angelegt und konfiguriert.
  → Siehe *Web Service aufrufen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.4, S. 413).*
- Sie haben eine SOAP-Nachricht für den Versand erstellt.
  → Siehe *Web Service aufrufen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.4, S. 413).*

So gehen Sie vor

1. Wählen Sie im Modul-Editor entweder den Web Services Medium/ Output oder Input Listener Connector, der die Binärdaten versenden soll.
2. Zeigen Sie für den jeweils gewählten Konnektor-Typ das Register „Erweitert“ an und fahren Sie beim entsprechenden Konnektor fort:
   - **Beim Medium/Output Connector**:
     a. Aktivieren Sie im Bereich „Anhänge/Binärdaten“ die Option „MTOM-Anhänge für ausgehende Nachricht aktivieren“.
     b. Passen Sie Ihre SOAP-Nachricht so an, dass die zu versendenden Binärdaten in einem Nachrichten-Element (z. B. mit dem Namen `imageData` für Grafikdaten) vom Typ `xmime:base64Binary` enthalten sind und weisen Sie dazu diesem Nachrichten-Element das Attribut `xmime:contentType` zu.
   - **Beim Input Listener Connector**:
     a. Aktivieren Sie im Bereich „Anhänge/Binärdaten“ die Option „MTOM-Anhänge für Antwort-Nachricht aktivieren“.
     b. Zeigen Sie das Register „XML-Schemas“ an.
     c. Wählen Sie das Nachrichten-Element, dessen Inhalt als Anhang verschickt werden soll (z. B. `imageData).`
     d. Weisen Sie diesem Element den Typ `xmime:base64Binary` zu, indem Sie diesen Typ aus dem unteren Bereich des Editors aus der Gruppe der „complex types“ auf das entsprechende Element oben in der Nachricht ziehen.

        Damit haben Sie dem Element gleichzeitig das notwendige Attribut `xmime:contentType` zugewiesen.

Beispiel für SOAP-Nachricht

3. Passen Sie die ausgehende SOAP-Nachricht so an (z. B. in einem XSLT Converter), dass im entsprechenden Nachrichten-Element, welches die MTOM-Daten enthalten soll, für das `xmime:contentType`-Attribut der korrekte Inhalts-Typ gesetzt ist (z. B. image/jpeg; im Zweifelsfall setzen Sie `application/ octet-stream` als generischen Inhalts-Typ, z. B.:

```
<soapenv:Envelope
xmlns:xmime="http://www.w3.org/2005/05/xmlmime">
 <soapenv:Body>
   <operationA>
     <imageData
       xmime:contentType:="image.jpeg/">
     </imageData>
     <pdfData
       xmime:contentType="application/pdf">
     </pdfData>
   </operationA>
 </soapenv:Body>
```

```
</soapenv:Envelope>
```

4. Legen Sie im Workflow folgende Variablen für die Daten des `imageData`-Elements und des `pdfData`-Elements an und definieren Sie deren Werte:
   - Variablen für die Daten des `imageData`-Elements:

Name	Typ	Wert
WSAttachment.0	`xs:base64Binary`	Bild-Daten
WSAttachmentName.0	`xs:String`	`imageData` (entspricht dem Namen des Nachrich-ten-Elements)

   - Variablen für die Daten des pdfData-Elements:

Name	Typ	Wert
WSAttachment.1	`xs:base64Binary`	PDF-Daten
WSAttachmentName.1	`xs:String`	`pdfData` (entspricht dem Namen des Nachrich-ten-Elements)

Die Variablen enthalten die Namen der zu referenzierenden Nachrichten-Elementen und die konkreten Daten.

→ Siehe auch
- *Abbildungsregeln für das Variablen-Mapping erstellen (Workbench: Benutzer-Guide, Kap. 14.8, S. 379)*
- *Variablen definieren (Workbench: Benutzer-Guide, Kap. 14.1, S. 366)*

Sie können einen File Connector zum Einlesen der Binärdaten aus einer Datei verwenden, den Sie im Workflow vor den Web Services Connector platzieren. Verwenden Sie dann ein Variablen-Mapping, in dem Sie die Binärdaten einer Variable vom Typ `xs:base64Binary` zuweisen.

40.13.2 Nachrichten mit binären Attachments empfangen

Standardmäßig werden Inhalte von Anhängen direkt in die empfangene SOAP-XML-Nachricht als base64-kodierte Daten integriert und dem Workflow übergeben.

Sie können den Inhalt von Elementen mit Binärdaten (bzw. die MTOM-Anhänge) auch extrahieren und als Inhalt von Workflow-Variablen setzen.

Voraussetzungen

Web Services Medium/Output oder Input Listener Connector ist angelegt und konfiguriert.

→ Siehe *Web Service aufrufen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.4, S. 413).*

So gehen Sie vor

1. Wählen Sie im Modul-Editor entweder den Web Services Medium/Output oder Input Listener Connector, der Binär-Inhalte oder MTOM-Anhänge empfängt.
2. Zeigen Sie für den jeweils gewählten Konnektor-Typ das Register „Erweitert“ an und fahren Sie beim entsprechenden Konnektor fort
   - **Beim Input Listener Connector**:
     Aktivieren Sie im Bereich „Anhänge/Binärdaten“ die Option „Binärdaten aus Eingangs-Nachricht extrahieren und als Variablen-Inhalte setzen“.
   - **Beim Medium/Output Connector**:
     Aktivieren Sie im Bereich „Anhänge/Binärdaten“ die Option „Binärdaten aus Antwort-Nachricht extrahieren und als Variablen-Inhalte setzen“.

Die eingehende SOAP-Nachricht wird nach Elementen mit dem Attribut `xmime:contentType` durchsucht und deren Inhalt wird in folgende Variablen geschrieben:

- `WSAttachmentName.[index]`: Name des Nachrichten-Elements
- `WSAttachment.[index]`: (Typ: `xs:base64Binary`) Daten des Nachrichten-Elements
- `WSAttachmentContent-Type.[index]`: Inhalts-Typ der Daten
- `WSAttachmentNumber`: Gesamtzahl der gefunden Inhalte

Wenn die Option „Anhänge im Dateisystem speichern" aktiviert ist, dann wird die Variable `WSAttachmentFileName.[index]` anstelle von `WSAttachment.[index]` gesetzt. Inhalt von `WSAttachmentFileName.[index]` ist der Name der Datei mit den Daten.

Um die in den Variablen ausgelesenen Daten an das nächste Modul zu übergeben, müssen Sie ein Mapping durchführen, welches die entsprechenden Variablen auf die Eingangsnachricht abbildet.

40.14 Dialogbeschreibungen

Dieser Abschnitt erläutert die folgenden Themen:

40.14.1 Dialog „Bereitgestellter Web Service"

(Input und Medium Listener Connector)

In diesem Dialog legen Sie fest, ob der Konnektor auf eine am Konnektor gespeicherte WSDL zugreifen soll oder auf eine WSDL, die im gesamten Workflow zur Verfügung steht.

WSDL-Definition

- **WSDL-Definition am Modul speichern**
  Wählen Sie diese Option, wenn die WSDL an dem Connector gespeichert werden soll. In diesem Fall steht die WSDL anderen, in demselben Workflow verwendeten Connectoren nicht zur Verfügung!
- **Am Workflow/PartnerLink gespeicherte WSDL-Definition verwenden**
  Wählen Sie diese Option, wenn die WSDL im Workflow gespeichert werden soll. Dann kann die WSDL auch von anderen, in demselben Workflow verwendeten Web Services Connectoren verwendet werden.
  Das Feld „Workflow" zeigt den Workflow-Namen an, sobald der Connector in einem Workflow verwendet wird.

Neue WSDL erzeugen

- **Callback-Listener-spezifische Einstellung**
  (nur Medium Listener Connector)
  - **WSDL von zugehörigem Aufrufer-Modul importieren**

Wählen Sie das entsprechende Modul aus, um den Callback-Listener zu konfigurieren.

- **Servicename**
  Dieses Feld ist mit dem Namen des Moduls vorbelegt.
- **Namensraum**
  Dieses Feld ist mit einem Namensraum vorbelegt, der aus dem Modulnamen erzeugt wird.
- **WSDL-Bindung**
  Zur Auswahl einer bereits vorhanden WSDL-Bindung oder zum Anlegen einer neuen WSDL-Bindung.
- **Typ**
  Zur Auswahl des SOAP-Messaging-Protokolls, an welches der Web Service gebunden wird.
- **Bindungs-Stil**
  Mit Ihrer Auswahl legen Sie die Struktur der SOAP-Messages fest:
  - **document/literal**
    Der Body der SOAP-Messages enthält ein XML-Element mit den Daten der SOAP-Message. Die Struktur dieses XML-Elements wird durch ein entsprechendes XML Schema-Element definiert.
  - **rpc/literal**
    Eine RPC-SOAP-Message entspricht einem Methoden-Aufruf mit Operationsnamen und einzelnen Parametern (mit simplen Datentypen).
    Der Body enthält den Operationsnamen aus dem Unterelement. Darunter sind die einzelnen Teile der WSDL-Message aufgeführt, deren Struktur ist durch XML Schema-Typen definiert.
  - **rpc/encoded**
    Entspricht dem „rpc/literal"-Stil mit dem Unterschied, dass der Typ der Nachrichtenteile explizit durch das Attribut `xsd:type` festgelegt wird.

Eine (englischsprachige) Diskussion der Vor- und Nachteile von WSDL-Stilen finden Sie unter http://www.ibm.com/developerworks/webservices/library/ws-whichwsdl/

- **Vorhandene WSDL übernehmen/integrieren**
  Aktivieren Sie diese Option, um eine bereits vorhandene WSDL zu nutzen. Die folgenden Optionen stehen Ihnen zur Verfügung:
  - **Als WSDL-Import hinzufügen**
    In die WSDL wird ein „wsdl:import"-Element mit der angegebenen URL als Referenz eingefügt.
  - **Inhalt übernehmen**
    Die Inhalte aus der im Feld „URL" angegebenen Datei werden in die WSDL kopiert.

- **URL**
  Mit einem Klick auf das Icon ▾ am Ende des URL-Feldes öffnen Sie ein Menü, um die URL der WSDL anzugeben, eine Repositorydatei oder ein Modul auszuwählen.

40.14.2 Dialog „Aufzurufender Web Service“

(Medium und Output Connector)

In diesem Dialog legen Sie fest, ob der Konnektor auf eine am Konnektor selbst gespeicherte WSDL zugreifen soll oder auf eine WSDL, die im gesamten Workflow zur Verfügung steht.

WSDL-Definition

- **Definition des aufzurufenden Web Services am Modul speichern**
  Wählen Sie diese Option, wenn die WSDL an dem Connector gespeichert werden soll.
- **Am Workflow/PartnerLink gespeicherte Definition verwenden**
  Wählen Sie diese Option, wenn die WSDL im Workflow gespeichert werden soll. In diesem Fall kann dieselbe WSDL auch von anderen, in demselben Workflow verwendeten Web Services Connectoren verwendet werden.
- **Workflow**
  Das Feld zeigt den Workflow-Namen an, sobald der Connector in einem Workflow verwendet wird.

40.14.3 Web Service Editor und Register

Dieser Abschnitt erläutert die folgenden Themen:

- *Register „Bereitgestellter Service“, S. 439*
- *Register „Aufzurufender Service“, S. 440*
- *Register „XML Schemas“, S. 442*
- *Register „WSDL Editor“, S. 442*
- *Register „Erweitert“, S. 442*
- *Register „SOAP-Nachrichten“, S. 448*

Der Web Service Editor wird nach dem Anlegen eines Web Services Connectors im Modul-Editor angezeigt, wenn Sie die Option „WSDL-Definition am Modul speichern“ gewählt haben.

40.14.3.1 Register „Bereitgestellter Service“

(Input Listener Connector)

In diesem Register erstellen Sie die Operationen, die Ihr Web Service anbieten soll.

Aus den Angaben werden automatisch eine WSDL-Datei als Schnittstellenbeschreibung Ihres Web Services sowie Ein- und Ausgangsnachrichten erzeugt.

- **Service-Name**
  Entspricht dem Namen des Connectors und kann manuell geändert werden.
- **Port-Name**
  Entspricht dem Namen des Connectors mit dem Suffix „Port“ und kann manuell geändert werden.
  Der Port definiert den Verbindungspunkt zu einem Web Service, vergleichbar dem Aufruf eines Moduls oder einer Klasse in einer traditionellen Programmiersprache.
- **Adresse**
  Adresse, unter welcher der Web Service angeboten wird. Ein Klick auf den ▼ Button öffnet folgendes Menü:
  - **Adresse kopieren**
    Kopiert die Adresse des Web Services in die Zwischenablage.
  - **Adresse der WSDL kopieren**
    Kopiert die Adresse der WSDL in die Zwischenablage.
  - **Publizierte Services anzeigen**
    Öffnet die Webseite http://<myServer>:8000/ibis/listServices und zeigt alle aktiven Web Services der inubit Process Engine an.

WSDL-Bindung

Name und Port-Typ der WSDL-Bindung sind änderbar, Typ und Bindungs-Stil sind nicht änderbar.

- **Name**
  Name der WSDL-Bindung.
- **Port-Typ**
  Name des WSDL-Elements „wsdl:portType“.

Diese beiden Felder müssen Sie nur dann anpassen, wenn Sie den Service umbenannt haben.

- Button „**Operation hinzufügen**“
  → Siehe *Operationen eines Web Services erstellen (Workbench/ Process Engine: Systemkonnektor-Guide, Kap. 40.8, S. 426).*

40.14.3.2 Register „Aufzurufender Service“

(Medium und Output Connector)

In diesem Register haben Sie folgende Optionen:

WSDL-Daten

- **WSDL laden von**
  - **Datei**
    Wenn Ihre WSDL-Datei in einem Dateisystem verfügbar ist, aktivieren Sie diese Option und geben Sie den Pfad der Datei an. Alternativ klicken Sie auf ➡, um zu der WSDL-Datei zu navigieren.
  - **URL**
    Wenn Ihre WSDL-Datei über eine URL verfügbar ist, geben Sie die URL der Datei an. Der ▾ Button am Ende des URL-Feldes öffnen ein Menü zur Auswahl folgender Funktionen:

Aktivieren Sie im Bereich „Web Service“ in demselben Dialog die Option „Authentifizierung erforderlich“, um bereits der Anfrage der WSDL Authentifizierungsinformationen mitzugeben.

    - **SSL**
      Öffnet einen Dialog, in dem Sie die Daten für eine sichere Verbindung angeben.
    - **Aus UDDI ermitteln**
      Um den Web Service über den UDDI Browser zu suchen und zu laden.
    - **Publizierte Services anzeigen**
      Öffnet eine Webseite mit einer Liste aller aktiven Web Services der inubit Process Engine.
  - **UDDI**
    Falls die WSDL-Datei in einer UDDI publiziert ist, wählen Sie die Option „UDDI“ und klicken Sie auf „UDDI Aufruf konfigurieren“, um den UDDI Browser zu öffnen.
    → Siehe *Dialog „UDDI Browser“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.14.5, S. 450).*
- **WSDL Datei laden**
  Lädt die angegebene WSDL in den Konnektor und zeigt sie an.

Editor/WSDL

In diesem Bereich wird die geladene WSDL angezeigt. Sie können diese bearbeiten, speichern und durchsuchen.

Web Service

- **Service/Port**
  Name und Port des aufzurufenden Web Services, der aus der Liste ausgewählt wurde
- **Adresse**
  Eine WSDL-Datei kann mehr als eine Adresse enthalten. Wählen Sie die Adresse aus, unter welcher der Web Service aufgerufen werden soll.
- **Button „SSL“**
  Öffnet den *Dialog „SSL-Konfiguration“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1.1.4, S. 24)* zum Konfigurieren einer sicheren Verbindung.
- **Aufruf stellt eine asynchrone Antwort (Callback) dar**
  Aktivieren Sie diese Option, wenn Sie diesen Web Services Connector in einem Workflow mit einem asynchronen Web Service anstelle eines Reply-Moduls verwenden wollen, um die Antwort an einen Callback Listener zu senden.

  In diesem Fall wird die Adresse aus der Variablen „WSReplyTo“ übernommen.

  → Siehe *Asynchronen Web Service anbieten (Workbench/ Process Engine: Systemkonnektor-Guide, Kap. 40.5.1, S. 418).*

Authentifizierung erforderlich

Zu markieren, wenn der Server, auf dem der Web Service veröffentlicht ist, eine Authentifizierung verlangt.

Falls diese Option aktiviert ist, werden die Authentifizierungsinformationen für den Service-Aufruf und für das Laden der WSDL verwendet.

- **Methode**
  Wählen Sie die gewünschte Methode für die HTTP-Authentifizierung aus. Standardmäßig ist die Authentifizierungsmethode „Basic Authentication“ eingestellt. Bei der Option „Automatisch (Digest/NTLM/Basic Authentication)“ wird per Nachrichtenaustausch die Authentifizierungsmethode zwischen Konnektor und Ziel-Server ermittelt.
- **Benutzername**/**Passwort**
  Wenn der Server eine Authentifizierung verlangt, geben Sie den Benutzernamen und das Passwort ein.
- **Domäne** (Nur bei Authentifizierungsmethode „Automatisch“)
  Geben Sie die Windows-Domäne für das NTLM-Verfahren ein.

Antwort asynchron empfangen/ separate Antwort-Adresse mitschicken

Aktivieren Sie diese Checkbox, wenn die Antwort des aufgerufenen Web Services asynchron an einen Callback Listener Connector gesendet werden soll.

- **Antwort-URL (‘wsa:ReplyTo‘)**
  Dieses Feld ist mit einer URL vorbelegt, die den Namen des Moduls, das Sie gerade anlegen, und das Suffix „Callback“ enthält.

Diese URL wird für die Kommunikation zwischen dem aufrufenden Modul und dem Callback Listener benötigt. Die URL entspricht der Service-URL des Callback Listener Connectors. Sie müssen die URL manuell anpassen, wenn sich die Service-URL des Callback Listener Connectors geändert hat.

40.14.3.3 Register „XML Schemas“

(Input Listener Connector)

Dieses Register zeigt alle XML Schemas an, die im Web Service verwendet werden.

Sie können vorhandene Schemas z. B. aus dem Repository importieren oder im Typ Editor neue Schemas erstellen.

→ Siehe
- *Editor verwenden (Workbench: Benutzer-Guide, Kap. 1.13, S. 54)*
- *Repositorydateien über URL ansprechen (Workbench: Benutzer-Guide, Kap. 19.10, S. 499)*

40.14.3.4 Register „WSDL Editor“

(Input Listener Connector)

Im Register „WSDL-Editor“ wird die WSDL-Datei angezeigt, die aus dem Web Service erstellt wird, der im Register „Bereitgestellter Service“ definiert ist.

Sie können diese WSDL bearbeiten oder eine vorhandene WSDL laden.

Wenn Sie eine vorhandene WSDL laden, werden fast alle Angaben in dem Register „Bereitgestellter Service“ überschrieben! Lediglich der Service- und Port-Name bleibt erhalten.

40.14.3.5 Register „Erweitert“

Proxy-Einstellungen

(Nur Medium und Output Connector)

Proxy-Einstellungen können Sie nur definieren, wenn Sie die Option „Kompatibilitätsmodus mit Version 5.0 (Einsatz des Axis2-Frameworks“ aktivieren. Wenn der Kompatibilitätsmodus aktiviert ist, werden folgende Bedienfelder angezeigt:

- **Globale Proxy-Konfiguration überschreiben**
  Wenn die Option nicht markiert ist, wird der Proxy-Server verwendet, der für die inubit Process Engine konfiguriert wurde.
  → Siehe *Proxy-Server zwischen inubit Process Engine und entfernten Rechnern konfigurieren (Process Engine: Administrator- und Entwickler-Guide, Kap. 5.3.2, S. 61).*
  Wenn die Option markiert ist, werden die folgenden Bedienfelder aktiviert und Sie können die globalen Proxy-Einstellungen überschreiben.
- **Servername/Port**
  Adresse und Portnummer des Proxy-Servers.
- **Domäne**
  Domäne, aus welcher der Zugriff über den Proxy-Server erfolgen soll.
- **Ausnahmen**
  Proxy-Einstellungen gelten nicht für die hier aufgeführten Systeme.
- **Authentifizierung erforderlich**
  Ist aktiviert, wenn der Proxy-Server eine Authentifizierung fordert.
- **Benutzername**
  Benutzername für den Proxy-Server.
- **Passwort**
  Passwort

Anhänge/Binärdaten

Mit diesen Optionen können Sie den Versand und den Empfang von Binärdaten, wie z. B. PDF- oder JPG-Dateien, als Anhang von SOAP-Nachrichten optimieren.

Die Optionen für den Umgang mit Binärdaten haben nur dann einen Effekt, wenn in der eingehenden bzw. ausgehenden SOAP-Nachricht ein Element mit dem Attribut `xmime:contentType` enthalten ist.

Siehe http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/

- **MTOM-Anhänge für ausgehende Nachricht aktivieren**
  (nur Medium/Output Connector)

Aktivieren Sie diese Option, wenn Sie in der Ausgangsnachricht den Inhalt von Elementen mit dem Typ `xmime:base64Binary` als MTOM-Anhänge verschicken möchten. Die Daten stammen aus `WSAttachment*`-Variablen.

→ Siehe *Nachrichten mit binären Attachments versenden (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.13.1, S. 432).*

- **Binärdaten aus Antwort-Nachricht extrahieren und als Variablen-Inhalte setzen**

  (nur Medium/Output Connector)

  Wenn diese Option aktiviert ist, wird die empfangene SOAP-Nachricht nach Elementen mit dem Attribut `xmime:contentType` durchsucht. Die Inhalte werden in `WSAttachment*`-Variablen geschrieben.

  → Siehe *Nachrichten mit binären Attachments empfangen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.13.2, S. 434).*

- **Anhänge ins Dateisystem speichern**

  (nur Medium/Output Connector)

  Wenn der aufgerufene Web Service Anhänge sendet, befinden sich diese in den entsprechenden MTOM-Variablen. Markieren Sie diese Option, um die Anhänge in das Dateisystem zu speichern. Die Pfade finden Sie in den MTOM-Variablen.

  Wenn die Option nicht markiert ist, dann bleiben die Anhänge in den Workflow-Variablen erhalten.

- **Binärdaten aus Eingangs-Nachricht extrahieren und als Variablen-Inhalte setzen**

  (nur Input Listener Connector)

  Wenn diese Option aktiviert ist, wird die empfangene SOAP-Nachricht nach Elementen mit dem Attribut `xmime:contentType` durchsucht. Die Elementinhalte werden in `WSAttachment*`-Variablen geschrieben.

  → Siehe *Nachrichten mit binären Attachments empfangen (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.13.2, S. 434).*

- **MTOM-Anhänge für Antwort-Nachricht aktivieren**

  (nur Input Listener Connector)

  Aktivieren Sie diese Option, wenn Sie in der Ausgangsnachricht den Inhalt von Elementen vom Typ `xmime:base64Binary` als MTOM-Anhänge verschicken (mit Daten aus „WSAttachment*“ Variablen) möchten.

  → Siehe *Nachrichten mit binären Attachments versenden (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.13.1, S. 432).*

Authentifizierung

(nur Input Listener Connector)

- **Authentifizierung erforderlich**

  Zu aktivieren, wenn der Service nur über eine HTTP-Basic-Authentifizierung zugänglich sein soll.
- **Benutzername**: Benutzername
- **Passwort**: Passwort
- **Bei Fehler eine Antwort mit Status 500 („Interner Server Fehler“) senden (anstelle von Status 401)**: Bei aktivierter Option verhält sich der Connector wie in früheren iS-Versionen und liefert einen HTTP-Status „500“ bei fehlgeschlagener Authentifizierung. Im Client muss in diesem Fall eine präemptive Authentifizierung eingestellt sein, das heißt, mit der Anfrage müssen die Authentifizierungsdaten mitgeschickt werden, anstatt auf eine HTTP-Status-401-Antwort zu warten.

W3C Standards

- **WS-Security**

  Zur sicheren Übertragung von Web Service-Nachrichten. Setzt die Authentifizierung am Service über Zertifikate voraus. Die Authentifizierung des Service-Konsumenten ist optional.

  - **Input Listener Connector**:

    Markieren Sie die Option, um WS-Security zu aktivieren. Zum Importieren der Zertifikate klicken Sie auf den Button „Einstellungen“.

    → Siehe *Dialog „WS-Security Konfiguration“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.14.7, S. 453).*
  - **Medium/Output Connector**:

    Sie erhalten beim Import der WSDL eine Meldung, falls der aufzurufende Web Service WS-Security verwendet. Klicken Sie auf den Button „WS-Security“, um die geforderten Zertifikate zu importieren.

    → Siehe *Dialog „WS-Security Konfiguration“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.14.8, S. 454).*
- **WS Reliable Messaging**

  (nur Input Listener Connector)

  - **Bei asynchroner Web Service-Kommunikation**

    Wenn markiert, dann wird das WS Reliable Messaging-Protokoll zur sicheren Übertragung der Web Service Calls an die inubit Suite 6 genutzt.

    Der erfolgreiche Empfang eines Requests wird bestätigt, sobald die Nachricht in dem Workflow angekommen ist, in dem der Input Listener Connector verwendet wird.
  - **Bei synchroner Web Service-Kommunikation**

Für die gesicherte Übertragung ist WS Reliable Messaging nicht erforderlich, weil der Empfang der Web Service Response den Empfang des Calls signalisiert.

Der Einsatz von WS Reliable Messaging ist trotzdem möglich, um die korrekte Abarbeitungsreihenfolge von Calls sicherzustellen.

→ Siehe *Dialog „WS Reliable Messaging Konfiguration“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.14.9, S. 455).*

Validierung

Die Validierung ist nützlich in Testphasen. Beachten Sie jedoch, dass die Validierung Performance kostet. Es kann daher sinnvoll sein, die Validierung im Produktivsystem abzuschalten.

- **Anfragenachricht validieren**
  - **Input Listener Connector**
    Wenn markiert, dann werden alle eingehenden Requests gegen die WSDL validiert.
  - **Output/Medium Connector**
    Wenn markiert, dann wird der ausgehende Requests gegen die WSDL validiert.
- **Antwortnachricht validieren**
  - **Input Listener Connector**
    Wenn markiert, dann werden die ausgehenden Responses gegen die WSDL validiert.
  - **Output/Medium Connector**
    Wenn markiert, dann wird die eingehende Response gegen die WSDL validiert.
- **Workflow starten trotz nicht valider Anfrage (Validierungsergebnis wird zum Workflow-Input)**
  (nur Input Listener Connector, nur im Kompatibilitätsmodus)
  Fügt das Ergebnis der Validierung der Ausgangsnachricht hinzu. Startet unabhängig vom Ergebnis der Validierung den Workflow und übergibt das Validierungsergebnis an das nachfolgende Modul.

Fehlerbehandlung

- **Bei Fehlerantwort Fehler werfen**
  Wenn markiert, dann wird ein Fehler bei der Workflow-Ausführung ausgelöst, wenn der Output Connector eine Fehlerantwort (Nachricht mit SOAPFault) empfängt.

Wenn der Fehler in der WSDL als mögliche Fehlerantwort definiert ist, bekommt der ausgelöste Fehler den in der WSDL angegebenen Namen.

Weitere Optionen

- **Automatische Anpassung der Service-Adresse in der WSDL deaktivieren**
  - Wenn die Option markiert ist:
    Der Host/Port-Teil der Service-Adresse in der publizierten WSDL wird nicht angepasst.
  - Wenn die Option **nicht** markiert ist:
    Der Host/Port-Teil der Service-Adresse wird aus der URL der WSDL-Anforderung übernommen. Die Service-Adresse in der publizierten WSDL wird wie folgt angepasst:
    - WSDL wird per `http://localhost:8000/ibis/ws/HelloWorld?wsdl` abgerufen
      In der WSDL steht die Service-Adresse `http://localhost:8000/ibis/ws/HelloWorld`
    - WSDL wird per `http://127.0.0.1:8000/ibis/ws/HelloWorld?wsdl` abgerufen
      In der WSDL steht die Service-Adresse `http://127.0.0.1:8000/ibis/ws/HelloWorld`
- **Kompatibilitätsmodus mit Version 5.0 (Axis2-Framework verwenden)**
  Aktivieren Sie diese Option, um sicherzustellen, dass Web Services Connectoren, die mit einer iS-Version < 5.1 angelegt wurden, auch in der aktuellen Version laufen.
  In der aktuellen Version wird Metro als Standard Web Service Framework eingesetzt, während in iS-Versionen < 5.1 das Axis2-Framework verwendet wurde.

UDDI

- **Publizieren**
  (nur Input Listener Connector)
  Zum Publizieren der WSDL-Datei in eine UDDI.
  Öffnet den *Dialog „UDDI Browser“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.14.5, S. 450)*.
- **Löschen**
  Zum Löschen einer bereits publizieren WSDL-Datei aus einer UDDI.

Namensräume

- **Namensräume bearbeiten**
  (nur Input Listener Connector)

Öffnet den *Dialog „Namensräume bearbeiten“ (Workbench/ Process Engine: Systemkonnektor-Guide, Kap. 40.14.10, S. 456)*.

40.14.3.6 Register „SOAP-Nachrichten“

Im Register „SOAP-Nachrichten“ können Sie aus den Web Services-Operationen im Register „Bereitgestellter Service“ bzw. „Aufzurufender Service“ eine SOAP-Eingabe- oder Ausgangsnachricht erzeugen und anzeigen lassen.

Diese Nachrichten können Sie z. B. als Templates im XSLT Converter verwenden.

40.14.4 Dialog „Web Service-Einstellungen“

Aufruf

Kontextmenü des zu konfigurierenden Web Service Connectors > Web Service-Einstellungen.

In diesem Dialog definieren Sie den aufzurufenden oder bereitgestellten Service, den Modus eines Listeners, die aufzurufende oder akzeptierte Operation und die Eingabe/Rückgabe-Daten.

Modus

(Input Listener Connector)

- **Request entgegennehmen und Workflow starten**
  Wählen Sie diesen Modus, wenn der Web Service asynchron aufgerufen wird und die Antwort über ein Reply-Modul gesendet wird.
- **Asynchronen Callback entgegennehmen und Workflow fortführen**
  Wählen Sie diesen Modus für einen Callback-Listener, an den ein Reply-Modul die Antwort asynchron sendet.

Service-Definition

In diesem Bereich wird die Adresse des bereitgestellten bzw. aufzurufenden Services angezeigt, die Sie im Modul-Editor definiert haben.

→ Siehe

- *Register „Bereitgestellter Service“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.14.3.1, S. 439)*
- *Register „Aufzurufender Service“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 40.14.3.2, S. 440)*

Mit einem Klick auf den -Button öffnen Sie ein Menü mit folgenden Optionen:

- **Adresse kopieren**
  Kopiert die Adresse des Web Services in die Zwischenablage.
- **Adresse der WSDL kopieren**
  Kopiert die Adresse der WSDL in die Zwischenablage.
- **Publizierte Services anzeigen**
  Öffnet die Webseite http://<myServer>:8000/ibis/listServices und zeigt alle aktiven Web Services der inubit Process Engine an.

Akzeptierte Operation

(Input Listener Connector)

- **Operation**
  Zur Auswahl der Operation, die der Web Service anbietet.

Empfangene Daten

(Input Listener Connector)

- **Ausgangsnachricht setzen**
  Wenn markiert, dann übergibt der Konnektor den empfangenen Request an den Workflow.
  - **Mit kompletter SOAP-Nachricht**
    Wenn markiert, dann wird eine vollständige SOAP-Nachricht ausgegeben.
  - **Nutzdaten (Inhalt von SOAP-Body)**
    Wenn markiert, dann wird nur der Inhalt des SOAP-Body ausgegeben.
- **Variablen-Inhalt setzen**
  Wenn markiert, dann schreibt der Konnektor den empfangenen Request in die angegebene Variable.
  - **Variable**
    Zum Auswählen einer bereits deklarierten Variable bzw. zum Anlegen einer neuen Variablen.

Aufzurufende Operation

(Medium/Output Connector)

- **Operation**
  Zur Auswahl der Operation, die der Web Service aufrufen soll.
  Die Operation ist standardmäßig nicht festgelegt. Es wird versucht, die Operation anhand der übergebenen SOAP-Nachricht zu ermitteln.
  - **Eigenschaften: anhand übergebener SOAP-Nachricht ermitteln**
    Die Option ist nur aktiviert, wenn Sie keine Operation festgelegt haben.

- **SOAP-Action**
  Geben Sie eine SOAP-Action an, wenn sie nicht aus der SOAP-Nachricht ermittelt werden kann oder soll.
- **Die aufzurufende Aktion liefert kein Ergebnis zurück.**
  Aktivieren Sie die Option, wenn die aufzurufende Aktion kein Ergebnis zurückliefert.

Eingabe/Rückgabe-Daten

(Medium/Output Connector)

- **Eingabe/Ausgabenachricht verwenden**
  Nur in Technical Workflows.
  Wenn markiert, dann wird der Web Service mit der Eingangsnachricht des Konnektors aufgerufen und die Rückgabenachricht (wenn vorhanden) wird wieder an den Konnektor übergeben.
  - **Mit kompletter SOAP-Nachricht**
    Wenn markiert, dann wird als Eingabe eine komplette SOAP-Nachricht mit Envelope und Body benötigt bzw. als Ausgabe eine komplette SOAP-Nachricht ausgegeben.
  - **Mit nur Nutzdaten (Inhalt von SOAP-Body)**
    Wenn markiert, dann wird als Eingabe nur das Element innerhalb des SOAP-Body benötigt bzw. in der Ausgangsnachricht nur der Inhalt des SOAP-Body ausgegeben.
- **Variablen verwenden**
  Wenn markiert, dann wird der Web Service Request aus der gewählten Eingabe-Variable erzeugt. Falls der Web Service eine Response zurück gibt, dann wird diese in die Rückgabe-Variable geschrieben.
  - **Eingabe-Variable**
    Zur Auswahl einer vorhandenen Variablen bzw. zum Anlegen einer neuen Variablen.
  - **Rückgabe-Variable**
    Zur Auswahl einer vorhandenen Variablen bzw. zum Anlegen einer neuen Variablen.

40.14.5 Dialog „UDDI Browser“

Der UDDI Browser steht auch im Web-Browser unter `http://<servername>:8000/uddi` zur Verfügung.

In diesem Dialog haben Sie folgende Optionen:

UDDI Verbindungskonfiguration

- **URL**
  Geben Sie die URL des UDDI-Registers ein oder wählen Sie ein UDDI-Register aus der Liste aus.
- **UDDI V2/UDDI V3**
  Zur Auswahl der UDDI-Version.

Suchen

Für die Suche nach Web Services in der UDDI-Register:

- **Name**
  Stichwort, nach dem die Inhalte der ausgewählten Kategorie durchsucht werden soll. Verwenden Sie `%` als Wildcard.
- **Suchen**
  Ein Klick auf den Button „Suchen“ startet die Suche.

Web Services, die mit der inubit Suite 6 erstellt und in eines der mitgelieferten UDDI-Register publiziert wurden, finden Sie, wenn Sie nach dem Namen des Web Services Listeners suchen.

Suchergebnis

Zeigt alle Web Services, die das gesuchte Stichwort enthalten. Markieren Sie einen Web Service, um ihn auszuwählen.

Ausgewählte URL

Zeigt den im Bereich „Suchergebnis“ ausgewählten Web Services an. Die Auswahl wird nach dem Klick auf „OK“ in das Eingabefeld übernommen.

40.14.6 Dialog „UDDI Dateneinstellungen“

(Input Listener Connector)

Um Ihren Web Service publizieren zu können, muss die Komponente „UDDI“ installiert sein.

→ UDDI ist ein Teil der inubit Suite 6. Siehe *Komponenten der inubit Suite 6 nachinstallieren (Process Engine: Administrator- und Entwickler-Guide, Kap. 3.3, S. 51).* Es werden die jUDDI Registry (2.0) und Nsure (3.0) von Novell installiert.

In diesem Dialog haben Sie folgende Optionen:

UDDI Verbindungskonfigurationen

- **URL**
  Wählen Sie eine UDDI Registry aus oder tragen Sie eine URL ein.
- **Authorization URL**
  Muss bei Nutzung einer UDDI 3.0 angegeben werden. Wenn Sie die mitgelieferte UDDI 3.0 nutzen, wird diese URL automatisch gesetzt.
- **UDDI V2/UDDI V3**
  Um die Version der ausgewählten UDDI anzugeben.
  Wenn Sie eine mitgelieferte UDDI ausgewählt haben, wird die korrekte Version automatisch angegeben.
- **Benutzername**
  Gültiger Benutzer der UDDI. Bei Auswahl einer mitgelieferten UDDI wird der Benutzername automatisch gesetzt.
- **Passwort**
  Gültiges Passwort der UDDI. Bei Auswahl einer mitgelieferten UDDI wird das Passwort automatisch gesetzt.

Web Service Interface publizieren

- **tModel publizieren**
  Wenn markiert, dann wird WSDL-Datei als tModel publiziert.
- **tModel Key**
  Wird nach dem Publizieren von der UDDI Registry zurückgeliefert.
- **Beschreibung**
  Geben Sie eine Beschreibung Ihres Web Service ein.

Web Service Implementierung publizieren

- **Business Service publizieren**
  Markieren Sie diese Option, um die WSDL-Datei als Business Service zu publizieren.
- **Service Key**
  Wird nach dem Publizieren von der UDDI Registry zurückgeliefert.
- **Business Key**
  Repräsentiert die Business-Gruppe, in dem der Service zur Verfügung steht.
- **Name**
  Service-Name der beim LateBinding als Suchparameter genutzt wird.
- **Beschreibung**
  Geben Sie eine Beschreibung Ihres Web Service ein.
- **Bindungsbeschreibung**
  Geben Sie eine Beschreibung der Bindungsart ein.
- **WSDL URL**
  URL des Web Service, die das UDDI beim LateBinding zurückgibt.
- **tModel Key**

Wird nach dem Publizieren von der UDDI Registry zurückgeliefert.

Siehe uddi.xml.org für weitere Beschreibungen.

40.14.7 Dialog „WS-Security Konfiguration“

(Input Listener Connector)

In diesem Dialog haben Sie folgende Optionen:

Service Authentifizierung

Die Authentifizierung des Service ist immer nötig, wenn WS-Security verwendet werden soll. Der Zertifikat-Handshake garantiert dem Service-Konsumenten, dass er den angeforderten Service erreicht hat. Die Zertifikate werden gleichzeitig zur Verschlüsselung der Nachricht verwendet.

- **Passwort**: Geben Sie das Passwort Ihres privaten Schlüssels an.
- **Keystore**: Zum Laden der `*.keystore`-Datei mit dem privaten Schlüssel. Sobald die Datei geladen ist, wird die Gültigkeit des Schlüssels angezeigt.

Consumer Authentifizierung

- **Sichere Session herstellen (WS Secure Conversation)**
  Stellt einen Security-Kontext her, der ein einmalige Authentifizierung ermöglicht: Wenn ein authentifizierter Service-Konsument den Service wiederholt aufruft, dann muss sich der Konsument nicht erneut authentifizieren.
- **Security-Mechanismus**
  Zur Auswahl stehen die folgenden:
  - **Service & Consumer authentication with XML encryption and signature**
    Fordert eine Authentifizierung vom Service und vom Konsumenten. Die Daten werden durch XML-Encryption verschlüsselt.
  - **Service & Consumer authentication with transport security (SSL)**
    Fordert eine Authentifizierung vom Service und vom Konsumenten. Die Daten werden durch SSL verschlüsselt.
  - **Only Service authentication with transport security (SSL)**
    Fordert eine Authentifizierung vom Service. Die Daten werden durch SSL verschlüsselt.
  - **Security Token Service issued Token with Service Certificate (STS)**

Um den Service durch einen STS abzusichern.

→ Siehe *Security Token Service Connector (Workbench/ Process Engine: Systemkonnektor-Guide, Kap. 33, S. 347).*

- **X.509/Datei auswählen**

  Wenn ein Security-Mechanismus mit „Consumer authentication“ gewählt wird, dann kann sich der Service-Konsument über X.509-Zertifikate authentifizieren.

  Der Button „Datei auswählen“ öffnet einen Dateiexplorer zum Speichern des Truststores mit den öffentlichen Schlüsseln der Service-Konsumenten. Ein Truststore hat die Dateiendung `.keystore`.
- **Benutzer/Passwort (UsernameToken)**

  Wenn ein Security-Mechanismus mit „Consumer authentication“ gewählt wird, dann kann sich der Service-Konsument über den Username authentifizieren.

  Mit den folgenden Optionen legen Sie fest, wie das Username Token verarbeitet werden soll:
  - **Interne Benutzerverwaltung**

    Das Username-Token wird gegen die interne Benutzerverwaltung der inubit Suite 6 validiert.
  - **Authentifizierung durch Workflow**

    Das Username Token wird innerhalb eines anzugebenden Workflows authentifiziert. Das Username Token-Passwort wird als Passwort-Moduleigenschaft verschlüsselt und dem Workflow als XML-Struktur übergeben.

    Die Authentifizierung wird als erfolgreich gewertet, wenn kein Fehler im Workflow geworfen wird.

40.14.8 Dialog „WS-Security Konfiguration“

(Medium und Output Connector)

In diesem Dialog haben Sie folgende Optionen:

Service Authentifizierung

X.509-Zertifikat: Button „Truststore oder Zertifikat auswählen“

Öffnet einen Dateiexplorer zum Laden des öffentlichen Zertifikats des Service. Wählen Sie eine `.cer`- oder eine Truststore-Datei. Wenn Sie einen Truststore wählen, werden Sie anschließend nach dem Alias der Zertifikats gefragt.

Consumer Authentifizierung

Zur Authentifizierung des Service-Konsumenten gegenüber dem Service.

- **Passwort**
  Passwort des privaten Schlüssels.
- **Keystore**: Button „Datei auswählen“
  Öffnet einen Dateiexplorer zum Laden der `*.keystore`-Datei, die Ihre privaten Schlüssel enthält.
- **Alias**
  Wenn mehrere private Schlüssel im Keystore liegen, müssen Sie den Alias des privaten Schlüssels angeben.

Username Authentifizierung

Name und Passwort des Kontos, das für den Zugriff auf den Server verwendet werden soll.

40.14.9 Dialog „WS Reliable Messaging Konfiguration“

In diesem Dialog haben Sie folgende Optionen:

- **Nachrichten in Reihenfolge des Versands verarbeiten**
  Die Option stellt sicher, dass Nachrichten in derselben Reihenfolge beim Empfänger ankommen, in der sie versendet wurden.
- **Nachrichten-Zustellsicherheit**
  Wählen Sie, wie oft versucht werden soll, eine Nachricht zuzustellen.
- **Ablaufsteuerung**
  Zum aktivieren oder deaktivieren der Nachrichten-Pufferung.
- **Maximale Puffergröße**
  Geben Sie an, wie viele Nachrichten maximal zwischengespeichert werden sollen. Ist die maximale Puffergröße erreicht, werden die übrigen Nachrichten ignoriert.
- **Timeout bei Inaktivität**
  Wenn der Service nach Ablauf der angegebenen Zeit keine Nachricht von dem Service-Konsumenten erhalten hat, dann beendet der Service die Übertragungssequenz wegen Inaktivität.

  Dasselbe gilt für Service-Konsumenten: diese beenden die Übertragung ebenfalls, wenn der Service nach Ablauf der angegebenen Zeit keine Nachricht gesendet hat.

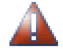
40.14.10 Dialog „Namensräume bearbeiten“

(Input Connector)

In diesem Dialog haben Sie folgende Optionen:

Ziel-Namensraum

Der Ziel-Namensraum umfasst alle Namensräume der WSDL.



Beachten Sie beim Ändern des Ziel-Namensraums, dass Sie auch bereits erzeugte SOAP-Nachrichten und Inhalte von Nachrichten-Variablen anpassen müssen!

Deklarierte Namensräume

Liste aller verwendeten Namensräume. Zum Hinzufügen, Bearbeiten und Löschen von Namensräumen.

41 WebDAV Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Mit dem WebDAV Connector binden Sie die inubit Process Engine an WebDAV-fähige Systeme an.

WebDAV (Web-based Distributed Authoring and Versioning) ist eine Erweiterung des HTTP-Protokolls, die es ermöglicht, mit mehreren Clients parallel Ressourcen und Collections auf WebDAV-Servern zu verwalten und zu bearbeiten.

- **Ressource**
  Ressourcen sind die auf dem WebDAV-Server verwalteten Dateien. Jede Ressource hat eine Menge von Eigenschaften (in XML notierte Properties), die über WebDAV-Methoden erfragt und geändert werden können.
- **Collection**
  Als Collections werden die Verzeichnis im Dateisystem des WebDAV-Servers bezeichnet. Jede Collection kann mehrere Ressourcen und Unter-Collections enthalten.

WebDAV definiert eine Reihe von HTTP-Methoden für die verschiedenen Protokollmechanismen. Anders als HTTP verwendet WEBDAV nicht ausschließlich HTTP-Header-Felder, um Requests und Responses mit zusätzlichen Parametern zu versehen, sondern kann bestimmte Parameter auch im Content-Body eines Requests übertragen.

Das Format der Eingangs- und Ausgangsnachrichten ist abhängig von der verwendeten WebDAV-Methode und den Anforderungen des WebDAV-fähigen Zielsystems.

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

41.1 Dialog „WebDAV Connector“

Einstellungen

- **URL**
  URL des WebDAV-fähigen Webservers.
- **SSL (Button)**
  → Siehe *Dialog „SSL-Konfiguration“ (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1.1.4, S. 24).*
- **Methode**

Erläuterungen zu allen WebDAV-Methoden finden Sie in der WebDAV-Spezifikation unter http://www.webdav.org/specs/rfc2518.html.

Geben Sie den Methodennamen ein oder wählen Sie eine WebDAV-Methode aus der Liste:

- `COPY`: Kopiert eine Ressource, der Name wird als URI im Request angegeben.
- `GET`: Fordert die im URI genannte Ressource an.
- `POST`: Sendet den Request-Body an den Server und erzeugt eine Ressource mit dem Namen, der als URI im Request angegeben ist.
- `PUT`: Erzeugt eine neue Ressource mit dem im Request angegebenen Namen.
- `MOVE`: Verschiebt eine Ressource, der Name wird als URI im Request angegeben.
- `DELETE`: Löschte die Collection oder Ressource, deren Name im Request angegeben ist.
- `MKCOL`: Erzeugt eine neue Collection mit dem im Request angegeben Namen.
- `PROPFIND`: Liest die Properties einer Ressource.
- `PROPPATCH`: Ändert und löscht mehrere Eigenschaften einer Ressource in einem atomaren Akt.
- `LOCK`:
  Reserviert eine Ressource oder eine Collection, sodass diese für den Schreibzugriff anderer Autoren gesperrt ist und exklusiv bearbeitet werden kann.
  Wenn der Request erfolgreich war, dann erhält der Client ein Lock-Token, das zum Aufheben der Sperre wieder vorgewiesen werden muss.
- `UNLOCK`:
  Entfernt die Sperre einer Ressource oder Collection, sodass diese auch von anderen Autoren bearbeitet werden kann.
- `SEARCH`:
  Löst eine Suche auf dem WebDAV-Server aus. Die Suchanfrage wird im Body des Requests definiert.

Bei Methoden wie z. B. `DELETE`, die keine Rückgabe liefern, wird eine Status-Response erzeugt, z. B.

```
<?xml version="1.0" encoding="UTF-8"?>
 <WebDAV version="1.0">
   <StatusCode>200</StatusCode>
 <StatusText>OK</StatusText>
</WebDAV>
```

- **Eingangsnachricht übertragen**
  Wenn markiert, wird zusätzlich zum Request die Eingangsnachricht zum WebDAV-Server übertragen.

Tabelle „Request Header/Wert“

Im Request Header können weitere Informationen in Form von Resource-Properties übermittelt werden, z. B.:

- Content-Type: Der MIME-Typ beschreibt das Format der übermittelten Daten, z. B. text/xml.
- From: Mailadresse des Anfragenden.
- Accept: Datenformate und Präferenzen des Benutzers für Daten. Enthält eine Liste von Daten- und Darstellungsformaten.

Um ein Property zu definieren, öffnen Sie das Kontextmenü der Tabelle und wählen „Hinzufügen“. Um den Wert des Propertys einzugeben, doppelklicken Sie in der Tabelle in die Spalte „Wert“.

Authentifizierung

- **Methode**
  Wählen Sie eine der beiden Methoden aus.
  - **Basic**: Wenn markiert, dann fordert der WebDAV-Server Benutzername und Passwort an, beides wird unverschlüsselt übertragen.
  - **NT**: Verwenden Sie diese Methode, wenn der WebDAV-Server auf einem NT-System läuft. Dieses Authentifizierungsschema ist ein Standard der Firma Microsoft und ermöglicht ein Single Sign-On auf Webservern unter Verwendung des Berechtigungsnachweises der Windows-Benutzeranmeldung.
    Wenn Sie diese Methode wählen, müssen Sie auch die Domäne angeben, an der Sie sich anmelden möchten.
- **Benutzername/Passwort**
  Zugangsdaten des Benutzers, mit dem Sie sich am WebDAV-Server anmelden möchten. Der Benutzer muss Zugriffsrechte entsprechend der ausgewählten WebDAV-Methode besitzen!
- **Domäne**
  Nur aktiv, wenn Sie die Methode NT gewählt haben. Name des Verwaltungsbereichs, an dem Sie sich anmelden möchten.

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

42 WebSphere MQ Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Der WebSphere MQ Connector verbindet die IBM WebSphere MQSeries Software mit der inubit Process Engine.

Konnektortypen

Abhängig von seiner Konfiguration sendet oder empfängt der Konnektor Nachrichten:

- **Input Connectors**
  Holt Nachrichten von einer Message Queue.
- **Input Listener Connector**
  Empfängt eine Nachricht von der WebSphere MQ Software, sobald neue Nachrichten in der Queue stehen, holt diese Nachrichten ab und startet den Workflow.
- **Output Connector**
  Schreibt Nachrichten in die Message Queue.

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

42.1 Voraussetzungen

Sie müssen der inubit Suite 6 die Datei `com.ibm.mq.jar` zugänglich machen.

So gehen Sie vor

1. Schließen Sie die inubit Workbench.
2. Stoppen Sie die inubit Process Engine.
3. Kopieren Sie die Datei `com.ibm.mq.jar` in das Verzeichnis `<iS-installdir>\client\lib\ext`.
   Die Datei ist Teil der IBM WebSphere MQ Software.

4. Kopieren Sie die Datei zusätzlich in eines der folgenden Verzeichnisse, abhängig vom eingesetzten Applikationsserver:
   - **Tomcat**:
     ```
     <iS-installdir>\server\Tomcat\webapps\ibis\WEB-INF\lib
     ```
   - **JBoss**:
     ```
     <iS-installdir>\server\JBoss\server\default\lib
     ```
5. Starten Sie die inubit Process Engine und die inubit Workbench neu.

42.2 Dialog „WebSphere Message Queue Connector“

In diesem Dialog haben Sie folgende Optionen:

Grundkonfiguration

- **Managername**
  Name des Queue Managers, zu dem die Verbindung aufgebaut werden soll.
- **Servername**
  Name des Servers, auf dem der WebSphere Application Server läuft.
- **Port**
  Nummer des Ports, über den die Kommunikation zum WebSphere Application Server stattfindet. Standardportnummer ist 1414. Der Button „Standard“ setzt jederzeit den Port wieder auf 1414 zurück.
- **Channel**
  Name des Channels ein, für den Sie die Verbindung konfigurieren wollen. Der MQ Series Queue Manager kann mehrere Channel haben. Jeder Channel muss einen eigenen Namen haben.

Queue Konfiguration

- **Queuename**
  Name der Queue, über die Sie Nachrichten empfangen möchten. In einem Channel können mehrere Queues verwaltet werden.
- **Methode**
  Methode zur Nachrichtenbehandlung. Wenn Nachrichten von einer Queue geholt werden, sind sie dort nach der FIFO-Methode gelistet. Die erste Nachricht, die empfangen wurde, wird als erste geholt.

- Input Connector
  - GET: Holt die erste Nachricht von der Queue und löscht diese aus der Queue.
  - BROWSE: Holt eine Kopie der ersten Nachricht aus der Queue. Die Originalnachricht bleibt in der Message Queue erhalten.
  - INQUIRE: Fragt die Anzahl der Nachrichten in der Queue ab. Das Ergebnis wird als ganze Zahl in einer XML-formatierten Nachricht ausgegeben.
- Output Connector
  `PUT`: Schreibt die Nachricht in die angegebene Message Queue.

Nachrichtenkonfiguration

- **Priorität**
  Standard ist `MQPRI_PRIORITY_AS_Q_DEF`. Dieser Wert wird aus dem Attribut `DefPriority` übernommen. Dieses Attribut ist Teil der Queue Manager-Konfiguration.
  Alternativ kann die Priorität mit ganzzahligen Werten zwischen 0 (geringste Priorität) und 9 (höchste Priorität) belegt werden.

Übernehmen Sie den Standardwert `MQPRI_PRIORITY_AS_Q_DEF`.

- **Format**
  Beschreibt, in welchem Format die eingehende Nachricht vorliegt. Der überwiegende Teil der Anwendungen, die Nachrichten über WebSphere MQ austauschen, benutzt eines der Formate MQFMT_STRING und MQFMT_NONE.
  - `MQFMT_STRING`: Zum Abholen von Nachrichten, die nur aus Zeichen besteht, also TextMessage, StreamMessage oder MapMessage.
  - `MQFMT_NONE`: Für ObjectMessage, BytesMessage und Nachrichten ohne Body.
  - `MQFMT_PCF`: (programmable command format) Für benutzerdefinierte Datenstrukturen.
- **Warteintervall**
  Wählen Sie beim Input Connector ein Warteintervall von 5 Millisekunden. Alternativ setzen Sie eine unbegrenzte Wartezeit (MQWI_UNLIMITED).

Übernehmen Sie immer den Standard 5!

Beim Output Connector hat diese Option keine Auswirkung.

- **Nur Nachrichten-Header lesen**
  (Nur beim Input Connector)

- Option „Zeilenweise empfangen“ ist nicht aktiviert:

  Die Ausgangsnachricht des MQ Connectors enthält die erste Nachricht der Message Queue. Zusätzlich werden die Header-Informationen dieser Nachricht als Modulvariablen ausgegeben.
- Option „Zeilenweise empfangen“ ist aktiviert:

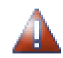
  Der MQ Connector liest zeilenweise aus der Message Queue bis zu einer Zeile mit dem String `#mqsende#`. Die Ausgangsnachricht besteht aus allen Zeilen bis zu diesem String. Die Header-Informationen werden der Nachricht mit dem String `#mqsende#` entnommen.

  Wenn keine der Nachrichten diesen String enthält, dann werden alle Zeilen aller Nachrichten aus der Queue gelesen. Die Header-Informationen stammen dann aus der letzten Nachricht und die Ausgangsnachricht enthält alle Zeilen aller Nachrichten aus der Message Queue.

- **Zeilenweise empfangen/versenden**:

  Überträgt die Nachricht zeilenweise, dabei wird jede Zeile einer Nachricht als eigene Nachricht behandelt und übertragen.

  Beim Versenden von Nachrichten in diesem Modus ist es wichtig, dass die letzte Zeile in der Nachricht der String `#mqsende#` ist. Dieser String gewährleistet beim Zeilenweise-Empfangen, dass nur die erste Nachricht in der Queue bis zum Ende gelesen wird. Wenn der String `#mqsende#` nicht enthalten ist, werden alle Nachrichten der Queue zeilenweise bis zum Ende der Queue gelesen.



Der MQ Connector fügt den String `#mqsende#` nicht ein. Sie müssen manuell sicher stellen, dass dieser String als letzte Zeile enthalten ist.

- **Nachricht zu XML konvertieren**:

Markieren Sie diese Option nur, wenn sichergestellt ist, dass jede Zeile, die vom MQ Connector gelesen wird, eine syntaktisch korrekte XML-Nachricht ist. Die XML-Nachricht wird durch einen internen Parser geprüft, bei einer fehlerhaften Nachricht wird ein Fehler erzeugt.

Erstellt Nachrichten, die alle gelesenen Zeilen enthalten. Jede Zeile ist eine XML-Nachricht mit dem Aufbau `<MQGET>Eine syntaktisch korrekte XML-Nachricht</MQGET>`.

- **Commit-Methode**: Das Commit stellt sicher, dass sich die Message Queue in einem konsistenten Zustand befindet.
  - **nach jeder Nachricht**: Das Commit wird nach jeder Nachricht durchgeführt. Diese Methode stellt sicher, dass nach jedem Abholen einer Nachricht der Queue Manager in konsistentem Zustand ist. Diese Option kann die Performance beeinträchtigen.

- **beim Verbindungsabbau**: Erst wenn alle Nachrichten eines Auftrags von der Queue geholt wurden, wird ein Commit durchgeführt. Diese Methode ist performanter.

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

43 X.400 SE Connector

Dieser Abschnitt erläutert die folgenden Themen:

Verwendung

Ein X.400 SE Connector empfängt und versendet Nachrichten auf Basis des internationalen Standards X.400.

Über X.400 werden Nachrichten in einem geschlossenen Rechnernetz ausgetauscht. Sowohl Absender als auch Empfänger der Nachricht besitzen X.400-Mailboxen in diesem Netz. Alle Nachrichten werden grundsätzlich über diese X.400-Mailboxen vermittelt, d. h. eine Nachricht wird zu der X.400-Mailbox des Senders geschickt, von dort an die X.400-Mailbox des Empfängers übermittelt und vom Empfänger abgeholt. In der inubit Suite 6 werden die Nachrichten von einem Isode-Client an die X.400-Mailbox übermittelt, der zusätzlich installiert werden muss.

Verbindungskosten reduzieren

Falls Ihr Provider die Verbindungen zu Ihrer X.400-Mailbox pro angefangene Minute oder pro Login berechnet, ist es sinnvoll, die folgenden Maßnahmen zu berücksichtigen, um die Verbindungskosten zu reduzieren:

- **Abholung von Nachrichten**:
  - Aktivieren Sie die Keep-alive-Funktion.
    → Siehe *Keep-alive aktivieren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 43, S. 471)*
  - Reduzieren Sie die Anzahl der Verbindungen pro Tag.
- **Versand von Nachrichten**:
  Bündeln Sie die Nachrichten vor dem Versenden.
  Konvertieren Sie dazu die Nachrichten in das IBISXML Format und führen Sie die Nachrichten mit einem XSLT Converter zusammen, bevor Sie diese versenden.
  → Das XML Schema und eine Beispielnachricht finden Sie im iS-Repository unter „`Global > System > Mapping Templates > X.400 SE Connector`".

Konnektortypen

Die Funktion eines X.400 SE Connectors ist von seiner Konfiguration abhängig:

- **Input Connector**
  Verbindet sich mit einer X.400-Mailbox, holt Nachrichten aus der Box ab und übergibt sie dem Technical Workflow.

Das Format der Daten, in welchem diese an den Workflow weitergegeben wird, hängt von der Konfiguration des Input Connectors ab.

- **Output Connector**

  Verbindet sich mit einer X.400-Box und versendet eine oder mehrere X.400-Nachrichten.

  Das Format der X.400-Nachricht hängt von der Konfiguration des Output Connectors ab.

→ Siehe auch

- *Systemkonnektoren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 1, S. 19)*
- *Module anlegen (Workbench: Benutzer-Guide, Kap. 3.3, S. 119)*
- *Technical Workflow Diagramme (Workbench: Benutzer-Guide, Kap. 12, S. 293)*

43.1 Voraussetzungen für den Betrieb

Zusätzliche Software

Sie müssen folgende Software installieren:

- **Isode-Client**

  Den Isode-Client erhalten Sie von der inubit AG.

43.1.1 Isode unter Windows als Dienst installieren

So gehen Sie vor

1. Installieren Sie den Isode-Client. Wählen Sie dabei das Setup „Typical“.
2. Um den Pfad zum `bin`-Verzeichnis des Isode-Clients der globalen `Path`-Variable hinzuzufügen, wählen Sie „Start > Einstellungen > Systemsteuerung“. Das Fenster „Systemsteuerung“ wird angezeigt.
3. Doppelklicken Sie den Eintrag „System“. Das Fenster „Systemeigenschaften“ öffnet sich.
4. Zeigen Sie das Register „Erweitert“ an.

5. Klicken Sie auf den Button „Umgebungsvariablen“. Das gleichnamige Fenster öffnet sich.
6. Markieren Sie im Bereich „Systemvariablen“ die Variable `Path`.
7. Klicken Sie auf „Bearbeiten“.
8. Fügen Sie am Ende der Einträge ein Semikolon und dann den Pfad zum `bin`-Verzeichnis des Isode-Clients ein.

43.2 Lesebestätigungen abholen und versenden

Lesebestätigungen versenden

Wenn für Nachrichten Lesebestätigungen angefordert sind, erzeugt und versendet ein X.400 SE Input Connector automatisch Lesebestätigungen:

- Nachrichten, für die eine Lesebestätigung angefordert wird, enthalten ein `isIPN`-Element mit dem Wert „false“.
- Lesebestätigungen enthalten ein `isIPN-Element` mit dem Wert „true“.

Lesebestätigungen abholen

Lesebestätigungen für Nachrichten, die mit einem X.400 SE Output Connector versendet wurden, werden nur dann aus der X.400-Mailbox abgeholt, wenn Sie XML-Nachrichten abholen.

43.3 Dialogbeschreibungen

Dieser Abschnitt erläutert die folgenden Themen:

43.3.1 Dialog „X.400 Zugangsdaten“

In diesem Dialog geben Sie die Zugangsdaten zu Ihrer X.400-Box und Ihre Absenderadresse an.

Grundkonfiguration

- **PA Adresse**
  Adresse des Providers Ihrer X.400 Box.
  Ersetzen Sie Hostnamen und Port durch die Angaben Ihres X.400-Providers.
  Hostnamen und Port-Nummer finden Sie am Ende der Vorbelegung: `URI+0000+URL+itots://securep7.telebox400.de:5432`
  - `securep7.telebox400.de`: Hostname
  - 5432: Port-Nummer

  Der Button „Standard“ stellt die Vorbelegung wieder her.
- **DN Bind**
  Domainangabe. Standard ist „C=de“ für Deutschland.
  Wenn Sie den X.400 SE-Konnektor in einem anderen Land einsetzen, müssen Sie das Länderkürzel entsprechend ändern.
  Der Button „Standard“ stellt die Vorbelegung wieder her.

Authentifizierung

- **OR Adresse**
  (Abk. f. Originator/Recipient) Absenderadresse)

Wenn Sie mehrere X.400 SE-Konnektoren mit derselben OR-Adresse konfigurieren, dann müssen alle Konnektoren dieselben Einstellungen haben. Sonst überschreiben sich die Konnektoren gegenseitig die Verbindungseinstellungen.

Beispiel: /S=INUBIT/O=INUBIT-IS/A=viaT/C=de.
Die folgende Tabelle erläutert die Adressbestandteile. Das beschriebene Format wird von der Deutschen Telekom unterstützt. Wenn Sie Ihre X.400-Box bei einem anderen Anbieter gemietet haben, erfragen Sie dort das unterstützte Adressformat:

Abk.	Steht für	Erläuterung	
S	Surname	Name der Firma oder Nachname der Absenders	Statt S und O können Sie auch die UA-ID verwenden, z. B. /UA-ID=2048188/A=viaT/C=de
O	Organization Name	Beliebiger Name, z. B. O=tu-berlin, wenn die Internetadresse cs.tu-berlin.telekom.de ist	

<table>
<tr><th>Abk.</th><th>Steht für</th><th colspan="2">Erläuterung</th></tr>
<tr><td>A</td><td>Administration Domain</td><td>Z. B. A=inubit, wenn die Internetadresse www.inubit.com ist. „viaT“ bedeutet, dass die Adresse bei der Telekom angemeldet ist.</td><td rowspan="2">Ob Angaben für A und C erforderlich sind und welche Werte sie enthalten, kann Ihnen der Empfänger mitteilen.</td></tr>
<tr><td>C</td><td>Country Name</td><td>Z. B. C=de für Deutschland</td></tr>
</table>

Groß- und Kleinschreibung werden nicht unterschieden.

- **Passwort**
  Sie erhalten das Passwort von Ihrem X.400-Provider.

Keep-alive-Konfiguration der Verbindung

- **Keep-alive aktivieren**
  Wenn markiert, dann wird die Verbindung zwischen X.400-Box und dem X.400 SE Connector so lange offen gehalten, bis das Timeout abgelaufen ist.
  Aktivieren Sie diese Option, wenn Ihr X.400-Provider die Kosten pro Login abrechnet.
- **Timeout (Sekunden)**
  Das Timeout gibt an, wie viele Sekunden nach der letzten Ausführung des Konnektors die Verbindung geschlossen werden kann. Durch Angabe eines Timeouts vermeiden Sie, dass Verbindungen z. B. nachts offen bleiben, obwohl keine Workflows ausgeführt werden.

Verbindungstest

- **Verbindung testen**
  Zum Testen, ob die Verbindung mit Ihren Angaben erfolgreich aufgebaut werden kann.

43.3.2 Dialog „Datenweitergabe konfigurieren“

(Input Connector)

In diesem Dialog konfigurieren Sie Filter und eine Lesereihenfolge für einzulesende Nachrichten sowie das Nachrichtenformat, das an das nächste Modul weitergegeben wird.

Nachrichtenfilter

- **Filter hinzufügen (Button)**
  Der Button fügt eine Filter-Option für Eingangsnachrichten ein.
  Sie können den Absender und den Betreff der abzuholenden Nachrichten mit den Bedingungen „enthält“/„enthält nicht“ filtern.
- **Nachrichten nach dem Lesen löschen**
  Wenn markiert, werden die Nachrichten nach dem Abholen aus der X.400-Box gelöscht.

Lesereihenfolge

Wenn mehr als eine Nachricht übertragen wird, legen Sie fest, welche Nachrichten zuerst übertragen werden und in welcher Reihenfolge.

Die Option „Alphabetisch“ sortiert nach der Betreffzeile, die Option „Chronologisch“ sortiert nach dem Datum.

Konfiguration des Ausgangsformates

- **Ausgabeformat**
  - **DATA**
    Die abgeholte Nachricht wird unverändert ausgegeben.
  - **XML**
    Die Nachricht wird in das IBISX400-Format konvertiert und ausgegeben.
    Das XML Schema sowie eine Beispielnachricht für dieses Format finden Sie im iS-Repository unter `Global > System > Mapping Templates > X.400 SE Connector`.
- **Nummer des Anhangs**
  (Nur bei Ausgabeformat „DATA“)
  Wenn mehrere Anhänge existieren, dann müssen Sie im Feld „Nummer des Anhangs“ angeben, welcher Anhang übertragen werden soll.

Beachten Sie, dass der Inhalt einer X.400-Nachricht als Anhang behandelt wird!

- **Max. Anzahl Nachrichten**
  (Nur bei Ausgabeformat „XML“)
  Maximale Anzahl von Nachrichten, die eingelesen werden.
  Falls ein Wert kleiner oder gleich Null gewählt wird, werden so viele Nachrichten abgeholt wie verfügbar sind.
- **Inklusive Anhänge**
  (Nur bei Ausgabeformat „XML“)
  Wenn markiert, dann werden auch Anhänge übertragen. Wenn nicht markiert, dann werden nur die Header-Informationen abgeholt.

Beachten Sie, dass der Inhalt einer X.400-Nachricht als Anhang behandelt wird!

- **Inhalte immer Base64 kodieren**
  (Nur bei Ausgabeformat „XML“)
  Um Binärdaten korrekt weiterzuleiten, z. B. als Zip- oder PDF-Dateien.
- **8-bit Text-Anhänge unkodiert abholen**
  Um Nachrichten ohne feste Zeichenkodierung als Raw-Daten weiterzuleiten. Die Nachrichten werden ohne Konvertierung aus dem Text-Anhang übernommen.
- **Nachrichten überspringen, die Fehler beim Abholen erzeugen**
  Wenn gesetzt, werden alle XML-Nachrichten, bei denen beim Abholen Fehler auftreten, ignoriert und auf dem Server belassen.
  Wenn nicht gesetzt, bricht das Abholen der Nachrichten mit einem Fehler ab.

43.3.3 Dialog „Nachrichtenoptionen für den Versand“

(Output Connector)

In diesem Dialog legen Sie u. a. Empfänger, Betreff und Format der zu versendenden Nachrichten fest und konfigurieren die Anhänge.

Um die Verbindungskosten zu senken, empfiehlt inubit AG, bei einer Ausführung des Konnektors mehrere Nachrichten zu versenden. Siehe *Verbindungskosten reduzieren (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 43, S. 467)*

Konfiguration des Eingangsformates

- **Format**
  - **TXT**
    Die Eingangsnachricht enthält Text in dem Format, dass bei der Option „Kodierung“ angegeben ist.
  - **DATA**
    Die Eingangsnachricht enthält Daten, die unverändert weitergegeben werden sollen.
  - **XML**
    Die Eingangsnachricht liegt im IBISX400 XML-Format vor.

Das entsprechende XML Schema sowie eine Beispielnachricht finden Sie im iS-Repository unter „`Global > System > Mapping Templates > X.400 SE Connector`“.

- **Kodierung**
  (Nur bei Text als Eingangsnachricht)
  Zeichensatz der Eingangsnachrichten. Die Angabe wird benötigt, um byte-kodierte Eingangsdaten korrekt in den Zeichensatz zu konvertieren, der bei der Option „Anhangtyp“ angegeben ist.

Eigenschaften der Nachricht

- **Betreff**
  Aussagekräftige Beschreibung der Nachricht.
- **Empfänger**
  → Für Infos über das Adressformat des Empfängers siehe *OR Adresse (Workbench/Process Engine: Systemkonnektor-Guide, Kap. 43, S. 470)*

„Betreff“ und „Empfänger“ sind optional, wenn Sie „XML“ als Eingangsformat ausgewählt haben. Wenn die Felder ausgefüllt sind, wird der Empfänger zusätzlich zu bereits in den Nachrichten vorhandenen Empfängern in die XML-Datei eingefügt. Der Betreff wird für alle Nachrichten übernommen und überschreibt die evtl. vorhandenen Betreffs.

- **Versandbestätigung**
  Wenn markiert, dann wird ein Nachweis darüber angefordert, dass die Nachricht verschickt wurde.
  Die Versandbestätigung wird automatisch vom X.400-Gateway an Sie als Absender der Nachricht gesendet und enthält die ID der verschickten Nachricht.
- **Lesebestätigung**
  Wenn markiert, dann wird eine Lesebestätigung angefordert.
  Die Lesebestätigung wird automatisch an Sie als Absender der Nachrichten gesendet, sobald die Geschäftspartner die Nachrichten aus der X.400-Box abholen.

Anhangskonfiguration

(Nur bei Eingangsformat „DATA“ und „TXT“)

- **Anhangtyp**
  Um festzulegen, mit welchem Zeichensatz die Eingangsnachricht des Output Connectors versendet werden soll, wählen Sie einen der folgenden Typen:
  - **ISO8859_1**
    8-Bit-Zeichensatz, enthält viele Sonderzeichen der westeuropäischen Sprachen.

    Siehe http://de.wikipedia.org/wiki/ISO_8859-1.
  - **ISO8859_2**
    8-Bit-Zeichensatz mit vielen Sonderzeichen der mittel- und südosteuropäischen Sprachen.

Siehe http://de.wikipedia.org/wiki/ISO_8859-2.

- **IA5**
  7-Bit-Zeichensatz gemäß ISO 646 mit verschiedenen lokalen Varianten. Die inubit Suite 6 unterstützt die Variante „IA5-US".

Siehe http://de.wikipedia.org/wiki/ISO_646.

- **Binary**
  Für Eingangsnachrichten im Binärformat.
- **FILE**
  Für Eingangsnachrichten im Binärformat mit FTBP-Anhang (File-Transfer-Bodyparts). Dieser Anhangtyp ermöglicht die Versendung zusätzlicher Dateiinformationen wie z. B. Dateiname.

Beachten Sie, dass dieser Anhangtyp nicht von allen X.400-Boxen unterstützt wird.

- **ISO8859_1_RAW**
  Für die Übertragung von Anhängen ohne feste Zeichenkodierung als Raw-Daten.

■ **Dateiname**
(Nur bei Anhangtyp=FILE)
Zusätzliche Angabe eines Dateinamens für die Ausgangsnachricht.